NASA/TM—2017-219500

A Compilation of MATLAB Scripts and Functions for MAC/GMC Analyses

Pappu L. N. Murthy and Brett A. Bednarcyk
Glenn Research Center, Cleveland, Ohio

Subodh K. Mital
University of Toledo, Toledo, Ohio

July 2017

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., "quick-release" reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 757-864-6500

- Telephone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Program
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM—2017-219500

A Compilation of MATLAB Scripts and Functions for MAC/GMC Analyses

Pappu L. N. Murthy and Brett A. Bednarcyk
Glenn Research Center, Cleveland, Ohio

Subodh K. Mital
University of Toledo, Toledo, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

July 2017

This work was sponsored by the
Transformative Aeronautics Concepts Program.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

This report is available in electronic form at http://www.sti.nasa.gov/ and http://ntrs.nasa.gov/

Contents

A Compilation of MATLAB Scripts and Functions for MAC/GMC Analyses

Pappu L. N. Murthy and Brett A. Bednarcyk
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Subodh K. Mital
University of Toledo
Toledo, Ohio 43606

Summary

The primary aim of the current effort is to provide scripts that automate many of the repetitive pre- and post-processing tasks associated with composite materials analyses using the Micromechanics Analysis Code with the Generalized Method of Cells (Ref. 1). This document consists of a compilation of several scripts that were developed in MATLAB (The Mathworks, Inc., Natick, MA) programming language and consolidated into 16 MATLAB functions (MAC/GMC). MAC/GMC is a composite material and laminate analysis software code developed at NASA Glenn Research Center (Refs. 2 and 3). The software package has been built around the generalized method of cells (GMC) family of micromechanics theories (Ref. 1). The computer code is developed with a user-friendly framework, along with a library of local inelastic, damage, and failure models. Further, application of simulated thermo-mechanical loading, generation of output results, and selection of architectures to represent the composite material have been automated to increase the user friendliness, as well as to make it more robust in terms of input preparation and code execution. Finally, classical lamination theory has been implemented within the software, wherein GMC is used to model the composite material response of each ply. Thus, the full range of GMC composite material capabilities is available for analysis of arbitrary laminate configurations as well.

The pre-processing tasks include generation of a multitude of different repeating unit cells (RUCs) for CMCs and PMCs, visualization of RUCs from MAC/GMC input and output files and generation of the RUC section of a MAC/GMC input file. The post-processing tasks include visualization of the predicted composite response, such as local stress and strain contours, damage initiation and progression, stress-strain behavior, and fatigue response. In addition to the above, several miscellaneous scripts have been developed that can be used to perform repeated Monte Carlo simulations to enable probabilistic simulations with minimal manual intervention. This document is formatted to provide MATLAB source files and descriptions of how to utilize them. It is assumed that the user has a basic understanding of how MATLAB scripts work and some MATLAB programming experience.

Nomenclature

L, H	Dimensions length, and height of repeating unit cell (RUC)
k_f	Fiber volume ratio (FVR)
k_i	Interface volume ratio (IVR)
Nrep	Number of RUCs for tiling
N_f	Number of fibers
d_f	Diameter of the fiber
t_i	Interface thickness
N_i	Number of subcells across the interface thickness
N_w	Number of subcells along width of the RUC

Nh	Number of subcells along height of the RUC
R_f	Radius of fiber
R_i	Outer radius of interface
d_c	Distance from center of subcell to center of RUC
x_2, x_3	Position of subcell
x_2^{new}, x_3^{new}	New position of mirrored subcell
k_m	Matrix volume ratio (MVR)
n_f	Number of fiber subcells
n_i	Number of interface subcells
$CovF$	Spread between the minimum and maximum fiber diameter
$CovI$	Spread between the minimum and maximum interface outer diameter
$Npix$	Number of pixels along width or height of RUC
VarI	Boolean to indicate whether the thickness of interface is constant
VarP	Fraction of interface thickness indicating the maximum variability of interface thickness
ε	Inter fiber clearance distance
$Eclr$	Fraction of fiber diameter by with the fiber clearance is calculated

1.0 Introduction

Composite micromechanics analyses based on the generalized method of cells (GMC) theory (Ref. 1) is utilized in developing a computer code (MAC/GMC) at NASA Glenn Research Center. This is well documented in References 2 and 3. The code can perform a comprehensive composite material and laminate mechanical analyses. This theory enables prediction of the local stress and strain fields in the composite material that are crucial for assessing damage initiation and progression in composite structures. The software package has been built around GMC to provide a user-friendly framework, along with a library of local inelastic, damage, and failure models. Further, application of simulated thermo-mechanical loading, generation of output results, and selection of repeating unit cell (RUC) architectures to represent the composite material have been automated in MAC/GMC. Finally, classical lamination theory has been implemented within MAC/GMC, wherein GMC is used to model the composite material response of each ply. Thus, the full range of GMC composite material capabilities is available for analysis of arbitrary laminate configurations as well. The many features that are available in the code, as well as the procedures to actually setup and run a problem, are well described in the MAC/GMC 4.0 User's Manuals (Refs. 2 and 3). The descriptions of the MATLAB scripts given below refer often to the MAC/GMC input and output file formats detailed in these documents.

The MAC/GMC software has been exercised with great detail in analyzing PMC and CMC composite materials by using nonresident (user defined) RUCs (ArchID 99 in the MAC/GMC input) over the past three years. The analyses included property predictions, stress-strain response compuations, fatigue and creep simulations and Monte Carlo based probabilistic simulations of both PMC and CMC materials using a variety of RUCs with number of fibers ranging from 1 to 100. The primary focus of these analyses was to assess the composite material microstructure variability and determine how it affects the properties and response of the overall composite. During the course of this work, a number of MATLAB scripts were developed to minimize the manual intervention and automate the analysis to a great degree. These routines can be categorized as (1) Preprocessing related tasks, (2) Automation of MAC/GMC runs, and (3) Extraction of desired properties/response of the composite material from MAC/GMC output. What follows is a compilation of all the MATLAB scripts in various categories with a brief description of the routine and its source code. The intention is to help future users by providing the source codes that can either be used as they are now or be easily modified to meet any specific requirements.

2.0 Preprocessing Related Tasks

These tasks include visualization of user defined RUCs, preparation of input blocks of data to be included in a typical MAC/GMC input file, extraction of RUC information from an existing MAC/GMC input file, and generation of a number of different types of RUCs that belong to the user defined RUC ArchID category 99 (Refs. 2 and 3). It should be noted throughout the document, RUC represents a two-dimensional matrix with elements being 1 or 2 or 3 where 1 represents fiber, 2 represents matrix and 3 represents an interface. These definitions are quite arbitrary and user can chose different numbers if desired but should maintain a consistency.

2.1 Plotting of 2-D RUCs With Uniform Subcells

The MATLAB routine is "PlotRUC(RUC)" which takes the definition of RUC in the form of a two-dimensional array matrix (wherein the elements are usually numbered 1 for fiber, 2 for matrix and 3 for interface for a Ceramic Matrix Composite (CMC) and 1 and 2 for a Polymer Matrix Composite (PMC) (as an interface is usually absent)) and creates a graphical representation of the RUC. The matrix RUC must be created either from an existing MAC/GMC input file or by using another MATLAB script. It should be noted here that the script uses one of the MATLAB internal functions "**`Imagesc`**" which comes with MATLAB image processing toolbox. To call the function use the syntax:

```
plotRUC(RUC);
```

where **`RUC`** is the matrix containing constituent information (1s, 2s, and 3s usually). This function is for a special case of an RUC where all subcells are square with same size. This enables us to use one of the MATLAB internal functions "**`Imagesc`**", which treats the RUC as an image and plots it as pixels.

As an illustration, the "***RUC**" block from a MAC/GMC input file, shown in Figure 1, is used to create Figure 2. In this example we only have two constituents: 1 for the fiber and 2 for the matrix. This is a user defined RUC (ArchID = 99) for analyzing a PMC material. The RUC size is 24×24.

```
*RUC
MOD=22 ARCHID=99
NB=24 NG=24
H=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
L=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,2,2,2,2
SM=2,2,2,2,2,1,1,1,1,2,2,2,2,2,2,1,1,1,1,1,1,2,2,2
SM=2,2,2,2,1,1,1,1,1,1,2,2,2,2,2,1,1,1,1,1,1,2,2,2
SM=2,2,2,2,1,1,1,1,1,1,2,2,2,2,2,1,1,1,1,1,1,2,2,2
SM=2,2,2,2,1,1,1,1,1,1,2,2,2,2,2,1,1,1,1,1,1,2,2,2
SM=2,2,2,2,1,1,1,1,1,1,2,2,2,2,2,2,1,1,1,1,2,2,2,2
SM=2,2,2,2,2,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,1,1,1,1,2,2,2
SM=2,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,2,2
SM=2,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,2,2
SM=2,2,1,1,1,1,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
```

Figure 1.—A typical "*RUC" block of a MAC/GMC input file where user choses an ARCHID 99, for a PMC material. The 1s and 2s form the numerical matrix, **RUC**, which is passed as an argument to the MATLAB script.

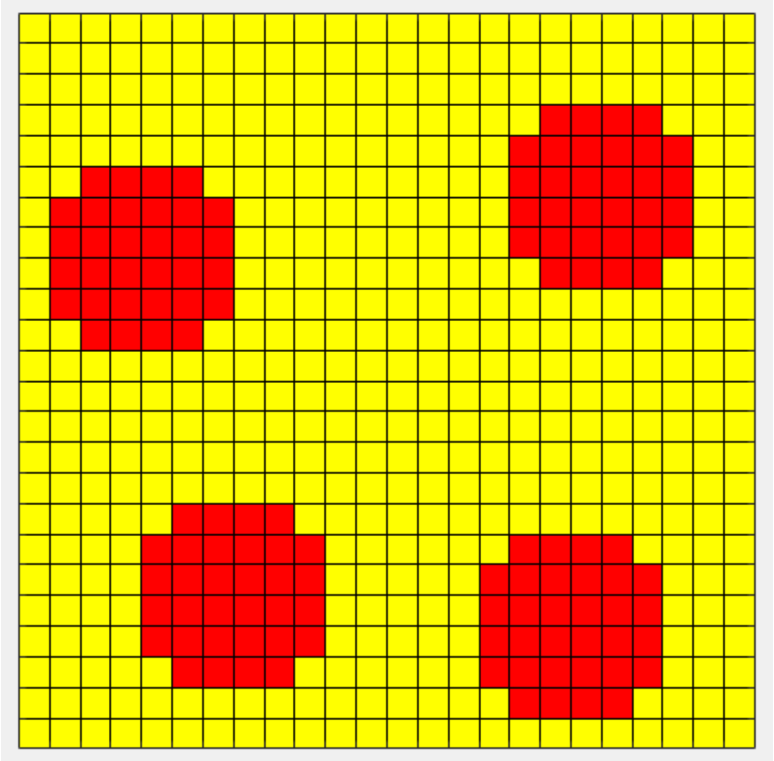


Figure 2.—A 24x24 RUC representing four randomly placed fibers with uniform subcells. No fiber/matrix interface material is present here.

The graphical output from the **plotRUC** function is shown in Figure 2. As seen, the RUC has four fibers which are placed randomly within the domain and is divided into 24×24 uniform square subcells. The fiber volume ratio is 0.22.

It should be noted that the current version of this function only supports up to three different constituents numbered 1, 2 or 3 only. Should the RUC contain more than three constituents, the function needs to be modified appropriately.

2.2 Plotting of 2-D RUCs With Nonuniform Subcells

The MATLAB routine "**ShowRuc**", which creates graphical plots of 2-D RUCs with nonuniformly shaped subcells, requires the input argument, **RUC**, as well as the vectors, **H** and **L**, which contain information regarding the height and length of each subcell as defined in the MAC/GMC manual (Refs. 2 and 3). To call the function, use the syntax:

```
showRuc(RUC,H,L);
```

where **RUC** is the matrix containing subcell location and type information (1s, 2s, 3s, and 4s usually). The '***RUC**' section of a typical MAC/GMC input file is shown in Figure 3. For more details regarding the MAC/GMC input file and the keywords, the user may refer to (Refs. 2 and 3). The RUC here is of size 17×17 (NB = 17 and NG = 17). The subcells are all nonuniform in size, and the RUC has three constituents; one for the fiber, two for the matrix, and three for the interface. A visualization of the RUC created using this input and the "**showRuc**" function is shown in Figure 4. Both fiber dimensions and interface thickness have simulated variability in this case.

```
*RUC
MOD=2 ARCHID=99
NB=17 NG=17
H=2.101124e-03,6.665675e-01,1.140424e-01,3.623440e-
01,9.356495e-02,9.698099e-02,5.042515e-02,8.019884e-
01,4.949958e-02,4.818913e-01,5.042515e-02,3.496777e-
01,4.949958e-02,2.431809e-01,9.356495e-02,4.613857e-
02,1.119413e-01
L=7.732065e-01,9.356495e-02,2.874894e-01,5.042515e-
02,7.934584e-01,4.949958e-02,1.222625e-01,9.356495e-
02,3.805714e-01,1.140424e-01,1.715528e-01,4.949958e-
02,4.455151e-01,9.637298e-03,5.042515e-02,5.398000e-
02,1.251384e-01
SM=2,3,1,1,1,1,1,3,2,3,3,3,3,3,3,3,2
SM=2,3,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2
SM=2,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,3,3,3,3,3,3,3,2,2,2,2,2
SM=2,2,2,2,2,3,1,1,1,1,1,3,2,2,2,2,2
SM=3,3,3,3,2,3,1,1,1,1,1,3,2,2,3,3,3
SM=1,1,1,3,2,3,1,1,1,1,1,3,2,2,3,1,1
SM=1,1,1,3,2,3,3,3,3,3,3,3,2,2,3,1,1
SM=1,1,1,3,2,2,2,2,2,2,2,2,2,2,3,1,1
SM=3,3,3,3,2,2,2,2,2,2,2,2,2,2,3,3,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2
SM=2,3,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2
SM=2,3,1,1,1,1,1,3,2,3,3,3,3,3,3,3,2
SM=2,3,1,1,1,1,1,3,2,3,1,1,1,3,3,3,2
SM=2,3,1,1,1,1,1,3,2,3,3,3,3,3,3,3,2
```

Figure 3.—A typical "*RUC" block of a MAC input file. The "**showRuc**" function requires not only the material layout as given by the **RUC** argument, but also the subcell dimension vectors as given by the **H** and **L** arguments.

As one might have realized, the more elaborate function "**showRuc**" may be used to plot any RUC and is not necessarily limited to those RUCs with nonuniform subcell sizes only as well those RUCs where H and L are of equal size. Furthermore, up to four different constituents can be specified in the RUC. However, the routine is much slower in rendering the graphics as each subcell needs to be drawn individually, unlike the "**Imagesc**" command that the "**plotRUC**" routine uses.

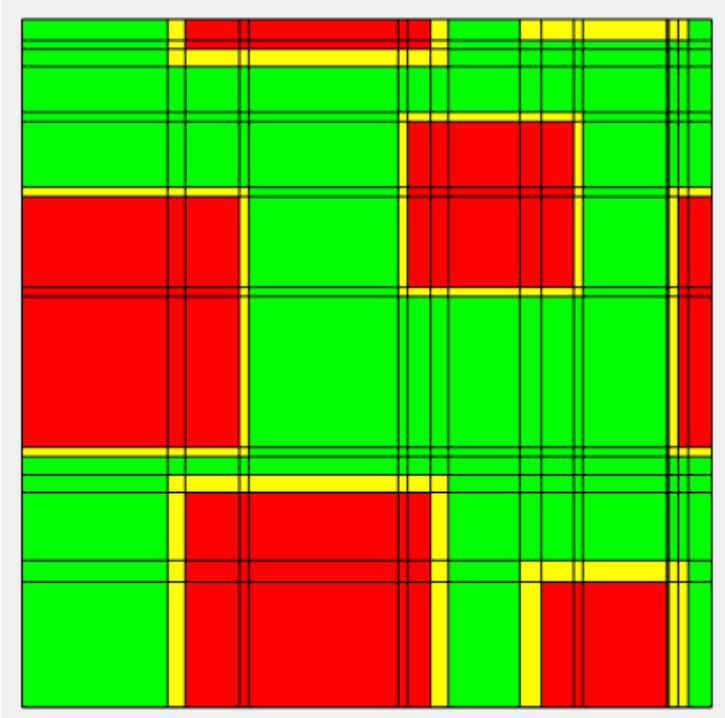


Figure 4.—A 17×17 RUC with fibers approximated as squares. The fiber size and the interface thickness are randomly varied.

2.3 Extracting RUC Information From a MAC/GMC Input File

In the above examples, we examined how to visualize a typical user defined RUC with both uniform and nonuniform subcell sizes. However, in order to utilize these routines, we need a definition for the **RUC** (usually a two-dimensional array matrix), as well as the **H** and **L** information, which are input arguments to the functions. The current function is developed to extract the required input information from an existing MAC/GMC input file as opposed to needing to manually entering the data. The function takes a typical '***RUC** input block' from an existing MAC/GMC input file, such as those shown in Figures 1 and 3, to extract the **RUC**, **H**, and **L** arguments. Note that one can use '***LAMINATE** input block' as well from a MAC/GMC input file to obtain the required information for the function. This routine is very useful when one needs to visualize an RUC based on an existing MAC/GMC input file. It should be noted here that one of the requirements for the function is that each H and L must be defined in one row. The RUC definition follows then where each row of subcells are described in one line starting with "**SM=**" followed by 1s, or 2s or 3s denoting various constituents. If continuations are used, for example in the case of very large RUCs to define any of the **H**, or **L** or "**SM=**" entries then the routine cannot be used as it assumes one line for each. In order to extract the RUC information, use the syntax:

```
[RUC,H,L] = ExtractSqRuc(f,Ln);
```

where **f** is the input file name. for example **f = 'Sample.txt'**, where **Sample.txt** is an ASCII file containing a ***RUC** block such as the ones shown in Figures 1 and 3. Ln is line number where the first 'SM=' input line starts. Note that this is usually 6 for **'*RUC'** and 7 for **'*LAMINATE'**. Rest of the input block is same in both cases. The function will extract and store the matrix **RUC** and the vectors **H** and **L** from the file so that they may be used as arguments in the functions **plotRUC** and **showRUC** described in Sections 2.1 and 2.2.

2.4 Writing of Any *RUC Block Input That Can Be Inserted Into a MAC/GMC Input File

Often user defined RUC information is generated using a separate MATLAB script, but once the arguments **RUC**, and **H** and **L** are generated, they need to be output as a suitable "***RUC** block" so that they may be inserted into a MAC/GMC input file. Such tasks often need to be performed when executing 100s of Monte Carlo simulations, wherein each instance has a randomly defined RUC microstructure generated on the fly. In such cases, the following routine may be utilized to translate and write the information contained in **RUC**, **H**, and **L** into a ***RUC** block. The following syntax should be used for writing the input block:

```
WriteRuc (RUC,FileIn,H,L);
```

Where **FileIn** is user-supplied ASCII text file name. It should be noted that the filenames have to be defined using quotes as shown below:

```
FileIn = 'Ruc.txt';
```

For very large RUCs, with sizes in excess of 200×200 subcells, it is recommended to use a slightly modified version of the above:

```
WriteBigRuc (RUC,FileIn,H,L);
```

This routine automatically limits each input line to have 200 entries and terminates with a MAC/GMC line continuation mark, "&". The remaining entries are deferred to the next line and so on. For example if we have a RUC with size 300×300 then H and L will have two lines of 200 followed by 100 entries each.

3.0 Creation of User Defined RUCs

This section focuses on how to create MAC/GMC RUCs using MATLAB scripts. These RUCs are based on (1) Square packing, (2) Hexagonal packing, (3) RUCs with random positioning of fibers, (4) RUCs with variable subcells sizes, and (5) RUCs to mimic a typical composite micrograph. A GUI based interface is recently developed for the generation of various RUCs and is documented in Reference 4.

3.1 Square Packing

RUCs that have circular fibers arranged in a square pack are probably most used of all the RUCs in MAC/GMC. The current routine creates an RUC with subcells that are perfect squares and a circular cross section fiber with a given fiber volume ratio and interface volume ratio (if an interface is present). Since discretization procedures with uniform subcell size involve round off errors, it is not possible to exactly satisfy the required fiber and interface volume ratios. In order to generate a square packed RUC with circular fiber use the syntax:

```
[fvr,mvr,ivr,RUC]=SquarePack(Fvr,Ivr,1,Ni,Nw);
```

where the user provided inputs are Fiber volume ratio (**Fvr**), Interface volume ratio (**Ivr**), number of subcells across the thickness of the interface (**Ni**) and the number of subcells in the width or height directions for a PMC material if the interface volume ratio is specified as '0'(**Nw**). In this case the value of Ni is automatically ignored. For PMCs the code determines the size of an RUC based on the value of Nw. A higher value of Nw is recommended to minimize the discretization error in the achieved Fiber Volume Ratio. Note that in case of a CMC RUC, the value provided for Nw is ignored automatically by the code.

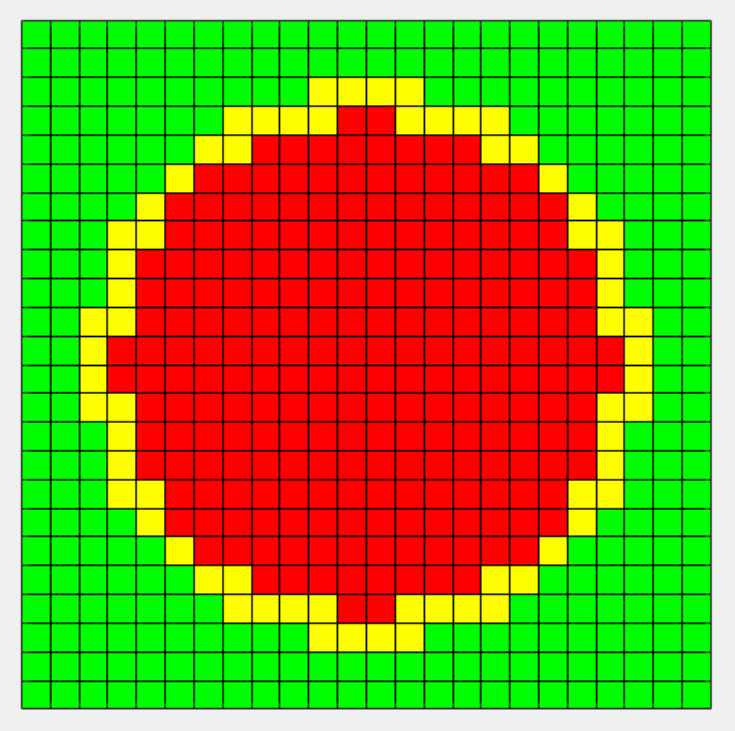


Figure 5.—A 24x24 CMC RUC with fvr = 0.4028, ivr = 0.1181 and mvr = 0.4792.

The size of the RUC depends on the value of Ni, the number of subcells in interface. After finalizing the RUC, the function will calculate the actual fiber volume ratio (fvr), interface volume ratio (ivr) and the matrix volume ratio (mvr). These may not agree exactly with **`Fvr`** and **`Ivr`** specified by the user due to discretization round off errors as mentioned before. Figure 5 shows a picture of a square packed CMC RUC. The user specified **`Fvr`** and **`Ivr`** are 0.4 and 0.1 in the example, however, the actual fvr and ivr realized are 0.4028 and 0.1181, respectively. The specified **`Ni`** is 1.

3.2 Hexagonal Packing

Hexagonally packed fibers are usually the recommended choice for representing the randomly distributed fibers in an actual composite system. RUCs that have circular fibers that are arranged in a hexagonal packing with uniform subcell discretization can be generated using the following syntax:

```
[fvr,mvr,ivr,RUC] = HexaPack (Fvr,Ivr,1,Ni,Nw);
```

where the user provided inputs are Fiber volume ratio (**`Fvr`**), Interface volume ratio (**`Ivr`**), number of subcells across the thickness of the interface (**`Ni`**) and number of subcells in width or height for a PMC material if the interface volume ratio is specified as '0' (**`Nw`**). In this case the value of Ni is automatically ignored. For PMCs the code determines the size of an RUC based on the value of Nw. A higher value of Nw is recommended to minimize the discretization error in the achieved Fiber Volume Ratio. Note that in case of a CMC RUC the value provided for Nw is ignored automatically by the code. The size of the RUC depends on the value of Ni, the number of subcells in the interface. After finalizing the RUC, the function will calculate the actual fiber volume ratio (fvr), interface volume ratio (ivr) and the matrix volume ratio (mvr). These may not agree exactly with **`Fvr`** and **`Ivr`** specified by the user due to discretization round off errors as mentioned before. Figure 6 shows a picture of a hexagonally packed CMC RUC. User required **`Fvr`** and **`Ivr`** are 0.4 and 0.1, however, the actual fvr and ivr realized are 0.3939 and 0.1061, respectively. The specified **`Ni`** is 1.

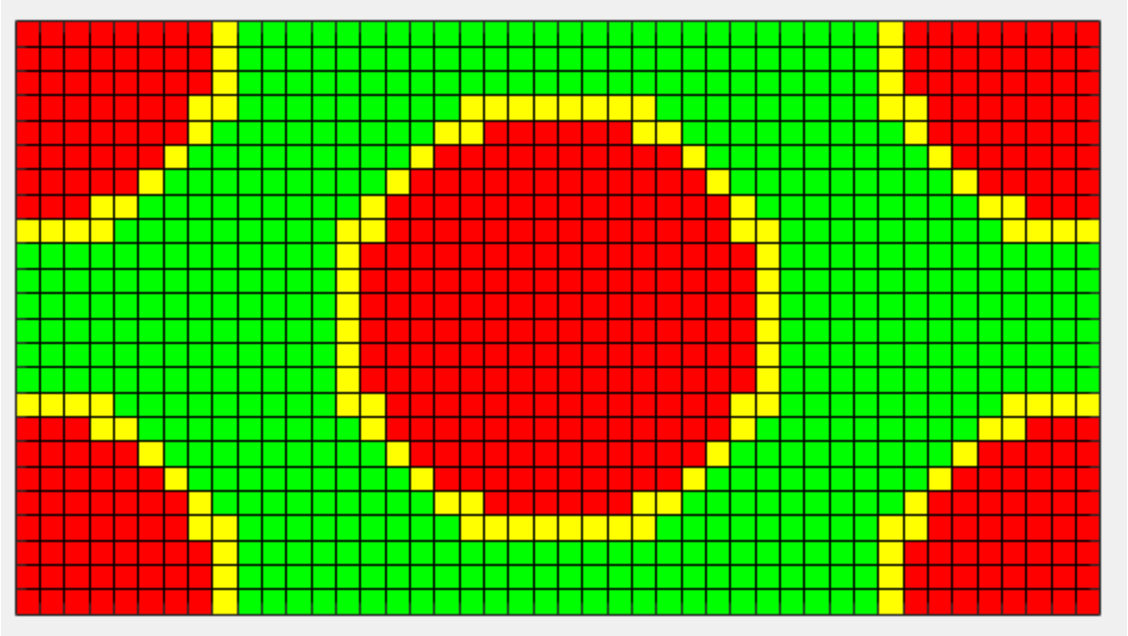


Figure 6.—A 24×44 CMC RUC with fvr = 0.3939, ivr = 0.1061 and mvr = 0.5.

3.3 Randomly Distributed Circular Fibers in Square RUC

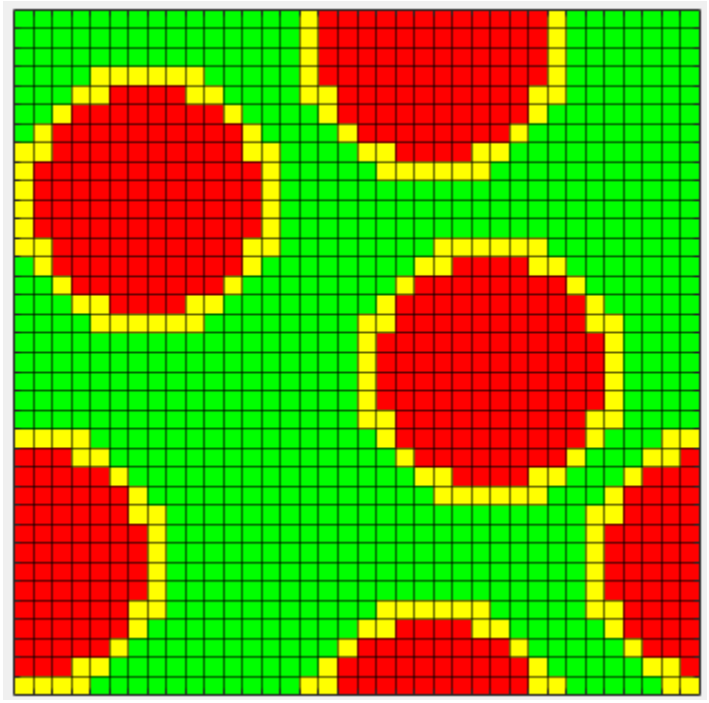
In order to represent randomly located fibers in an actual composite microstructure, RUCs can be created wherein the fiber locations are randomly chosen. Furthermore, whenever a fiber crosses an RUC boundary, the remaining portion of the fiber has to be located by mirroring it on the opposite side of RUC. Thus, when the RUCs are tiled, we get whole fibers everywhere. It should be noted that the periodicity conditions that are applied by the Generalized Method of Cells theory (Ref. 1) can only be properly enforced when we impose the mirroring conditions. The method behind this script is quite complicated and often many trials are required before a successful configuration can be found. The code automatically performs these trials based on a preset parameter for number of trials in the code which is usually a very high number(10000). A specialized algorithm is developed where in once a fiber with interface is placed all the subcells and the neighboring subcells where another fiber cannot be placed because of fiber intersections, are fenced out for the subsequent fiber placements. The subsequent fibers are chosen to be placed randomly from the list of remaining available subcells. This process continues till all the fibers are placed or when there are no remaining available cells. The process repeats with a fresh start till all the fibers are successfully placed. For PMC's with interface one can achieve as high as 70 percent fvr, allowing fibers to touch. The user must exercise caution when generating such RUCs by providing fiber and interface volume ratios that are reasonable such that an acceptable RUC can be generated in a reasonable number of tries. To generate an RUC with randomly located fibers the user should use the syntax:

```
[FVR,IVR,MVR,RUC,BigRUC,BigL,BigH,FibCenters,Ta] =...
  new_CMC_Ruc2 ( RadF,RadI, Nfibers,nrep,dpad,MaxTries,fvr,ClCntrl,ClExpo)
```

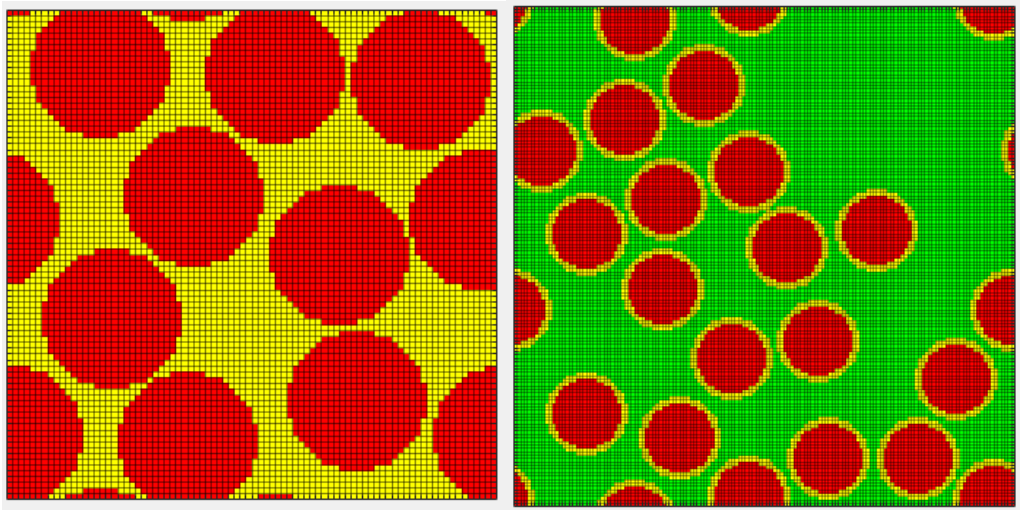
The user must provide values for radius of fiber (**`RadF`**), and the outer radius of fiber with interface (**`RadI`**) in terms of number of subcells. Additional inputs to the script are the fiber volume ration desired (fvr), and number of fibers to be placed in RUC (**`Nfibers`**). The two additional parameters ClCntrl and ClExpo are advanced features than enable to control fiber clustering with an RUC when the fiber volume ratios are low and there are a large number of fibers desired to be placed. ClCntrl = 0, sets the normal alogorithm and to invoke the cluster control one should set ClCntrl = 1 and give a value usually around 6 for ClExpo. ClExpo is the exponent value that controls the degree of attraction between the fibers to cluster together. Higher values ensure greater the cluster attraction. For example, if the user choses

RadF = 6, RadI = 7, fvr = 0.35, Nfibers = 4, the code generates an RUC as shown in Figure 7(a). The actual realized fiber volume ratio (FVR) = 0.3457, the matrix volume ratio (MVR) = 0.5185 and the interface volume ratio (IVR) = 0.1358. The RUC size is 36×36.

In Figure 7(a), the fiber at the top is cut off, and the remaining portions of is mirrored on to bottom edge. Similarly, one of the fibers is cut off by the left edge and the remaining portion is shown near the right edge. If a fiber is cut off at a corner, then all four corners will be filled appropriately in order to preserve the periodicity of the RUC. Note in this example the cluster control parameter is set to 0. In order to illustrate how extremely high fiber volume ratios can be realized, we show here two cases one with no clustering and the other with a high degree of clustering, in Figure 7(b).



(a)



(b)

Figure 7.—(a) A 36x36 CMC RUC with FVR = 0.35, IVR = 0.14 and MVR = 0.51. (b) Left: Example of a PMC RUC (84×84) of 10 fibers with FVR=0.635. Right: Example of fiber clustering in CMC. RUC (145×145) for 20 fibers with a FVR of 0.3 and IVR of 0.126.

3.4 Randomly Distributed Square Shaped Fibers in Square Packing

This is a unique case of an RUC where the generated subcell size can be nonuniform. The fibers are approximated by a square shape instead of the customary circular shape. An example of such an RUC is already seen in Figure 3. A major advantage of such a representation of an RUC is that the size of the RUC is much smaller than the corresponding circular fiber case with unform subcells. For example, the RUC size is always equal to (**Nfibers×4 + 1** by **Nfibers×4 + 1**). So for a four fiber case, the RUC will be of 17×17 size and so on. Furthermore, the fiber and interface volume ratios can be met exactly as specified by the user. Last, when analyzing the RUC using MAC/GMC's high-fidelity generalized method of cells (HFGMC) (Ref. 1) option, the computational time involved for such an RUC is sometimes even an order of magnitude lower than the corresponding circular fiber case. The advantages and disadvantages in using such RUC representations for CMCs are well documented in recent inhouse studies where the composite stiffnesses, first ply matrix strength, and fatigue performance are assessed using square and circle representations of fibers in addition to using the GMC and HFGMC modeling options (Refs. 5 to 7).

In order to generate a randomly distributed square shape fibers in square packed RUC the following syntax is needed:

```
[RUC,L,H] = ...
RectFiberPlacementWithMirroring(Nfibers,fvr,ivr,Eclr,CovF,CovI);
```

where **fvr**, **ivr**, and **Nfibers** are the fiber volume ratio, the interface volume ratio and the desired number of fibers specified by the user. The parameter **Eclr** is usually taken as 0.01 and represents percentage of fiber typical diameter and is used to maintain a minimum clearance between the fibers so that fibers do not touch each other. The parameters **CovF**, and **CovI** are the coefficients of variation in fiber size and interface thickness size and need to be specified as fractions. These are utilized in the generation of variable size fibers and interface thicknesses. The outputs are the **RUC** matrix and the **L** and **H** vectors. The function utilizes four other functions 1) **CheckConstituent**, 2) **IntersectFibers**, 3) **Grid** 4) **BdCheckTest**. The function **CheckConstituent** is called to identify constituent material of each subcell. The function **IntersectFibers** is called to check whether any of the fibers generated so far intersect each other. If the routine finds any intersection, the configuration is discarded and a new configuration is generated. The function **Grid** is utilized to determine the **H** and **L** vectors containing the subcell dimension information once all the fibers are placed within the RUC. The function **BdCheckTest** is called to check whether any of the newly placed fibers intersect the boundaries of the RUC. In the case of any intersection, the program automatically cuts off the fiber outside of the RUC boundary and places it on the opposite side using mirroring.

As an example the following command will generate 5 fibers in an RUC with a fvr of 35 percent, ivr of 10 percent and CovF and CovI = 0.20;

```
[RUC,L,H] = RectFiberPlacementWithMirroring(5, 0.4, 0.1, 0.01, 0.2,
0.2);
```

Figure 8 shows the generated RUC. The fibers show a 20 percent variability in size, while the interface thickness varies with a 20 percent coefficient of variation as well. The RUC has a size (21×21). To show the RUC we used the command:

```
showRuc(RUC,0,0,H,L);
```

As described in Section 2.2.

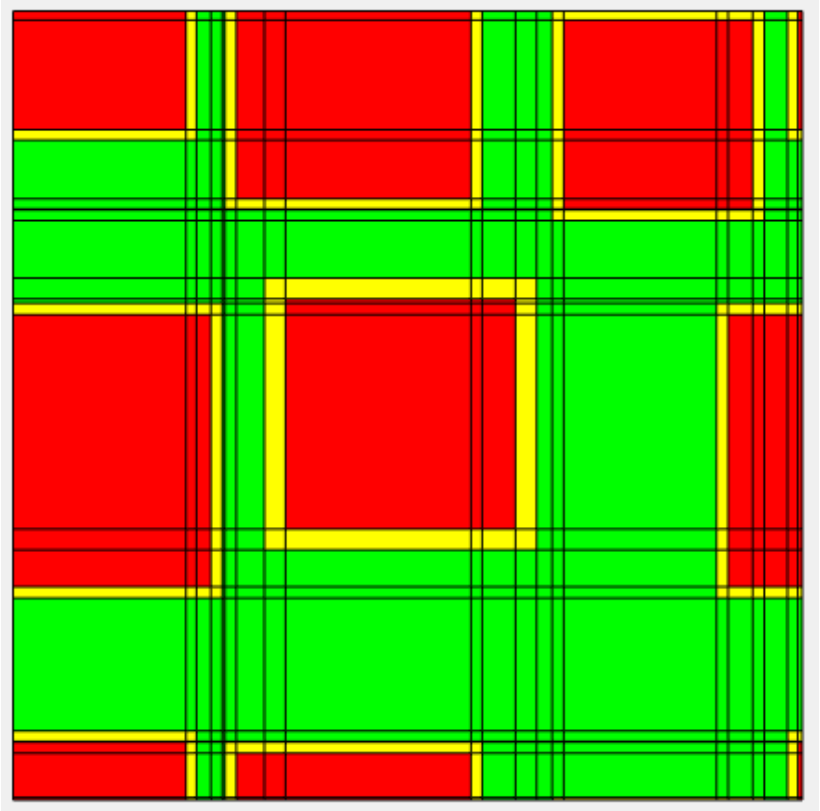


Figure 8.—A 21x21 CMC RUC with fvr = 0.35, ivr = 0.1, and mvr = 0.55. Fibers are approximated as squares of variable sizes.

3.5 Generation of a PMC RUC With Randomly Placed Fibers

When analyzing composite micrographs for PMC materials, sometimes it is required to model with a reasonable RUC size a typical PMC micrograph with tens to hundreds of fibers. Here, in order to keep the size practical, one may have to approximate each fiber as a single subcell of square dimensions and with a given fiber volume ratio but with random distribution of the fibers throughout the RUC. To generate such an RUC the function '**RandRuc**' may be utilized with the following syntax:

```
[RUC,H,L,fvr,mvr] = RandRuc(FVR,nH,nL);
```

Where **FVR** is the desired fiber volume ratio, **nH** and **nL** are the number of subcells in the height and length directions of the RUC, respectively. The function outputs **RUC** matrix, the **H** and **L** vectors, and the realized fiber and matrix volume ratios are **fvr** and **mvr**. To generate the RUC shown in Figure 9, the following command is used:

```
[RUC,H,L,fvr,mvr] = RandRuc(0.6,20,15);
```

The realized fiber volume ratio (**fvr**) is 0.616 and the realized matrix volume ratio is 0.384. It should be noted that such RUCs are usually analyzed to obtain the gross stiffness related properties in general. The localized stress and strain fields obtained using such models are likely not accurate.

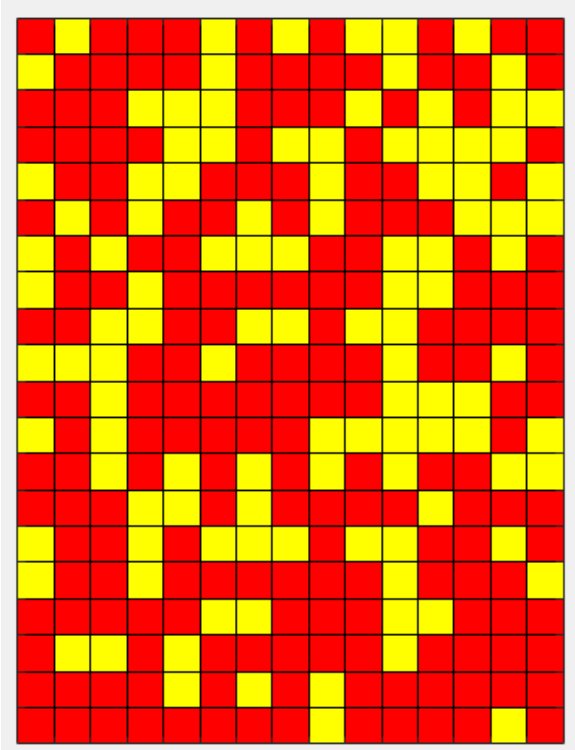


Figure 9.—A 20x15 PMC RUC with fvr = 0.616, mvr = 0.384. Red subcells are fibers, and yellow subcells are matrix.

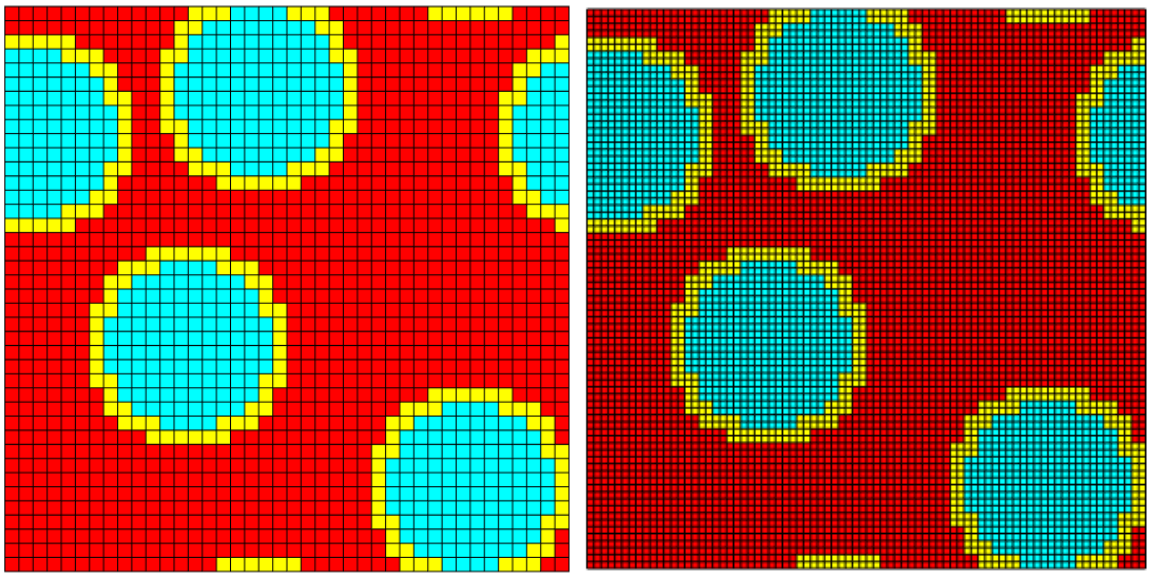
3.6 Increasing Subcell Density Without Affecting the Shape of Fiber in a Self Similar Fashion

Sometimes it becomes necessary to increase the RUC size by doubling or tripling the number of subcells in the H and L directions in order to study factors such as mesh density sensitivity. Once a basic RUC with uniform sub cells is generated, a higher density RUC can be generated using the function HDRuc. This can be done with the following command line script;

```
[RUC2] = HDRuc(RUC,density)
```

in which the above RUC is the parent RUC whose subcell to be increased. The variable density can take integer values only such as 1, 2 or 3. Each subcell of the parent RUC is divided into 1×1, or 2×2 or 3×3 subcells and the high density version of the RUC is output as RUC2.

An example of a 40×40 subcell RUC which is increased to a 80×80 subcell model is shown pictorially in Figure 10. The steps to achieve this are given below:



Size = 40x40 Size = 80x80

Figure 10.—In the above the original RUC with 40x40 subcells is modified to RUC2 with 80x80 subcells by choosing a density = 2.

The above is achieved by using the commands

```
[RUC,H,L] =  ExtractSqRuc('UserRUC40.txt')
```

Which extracts the base RUC of size 40×40 from the file "UserRUC40.txt".shown in Figure 11.

```
% Increase Density
density =2;
[RUC2] = HDRuc(RUC,density)
H2 = repmat(H,1,density); % Increase the Height dimension
L2 = repmat(L,1,density); % Increase the Length dimension
```

Where the high density repeating unit cell RUC2 has a size of 80×80. H2 and L2, respectively the Height and Length of RUC in terms of subcells. The higher density repeating cell can be shown using the following command:

```
showRuc(RUC2,H2,L2,[0 1 1;1 0 0;1 1 0])
```

```
*RUC
MOD=2 ARCHID=99
NB=40 NG=40
H=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
L=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=3,3,3,3,3,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3
SM=1,1,1,1,3,3,3,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3
SM=1,1,1,1,1,1,3,3,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1
SM=1,1,1,1,1,1,1,3,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1
SM=1,1,1,1,1,1,1,3,3,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1
SM=1,1,1,1,1,1,1,1,3,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1
SM=1,1,1,1,1,1,1,1,3,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1
SM=1,1,1,1,1,1,1,1,3,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1
SM=1,1,1,1,1,1,1,1,3,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1
SM=1,1,1,1,1,1,1,3,3,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1
SM=1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1
SM=1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1
SM=1,1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3
SM=3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
SM=2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2
SM=2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2
SM=2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2
SM=2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2
```

Figure 11.—“*RUC” block of a MAC/GMC input file where user choses an ARCHID 99, for a CMC material. The size of RUC is 40x40.

4.0 Post Processing Related Tasks

MATLAB functions have also been generated to accomplish MAC/GMC post-processing tasks including extraction of response quantities from a MAC/GMC analysis output file such as material stiffness related properties, laminate properties, local stress and strain response contours, fatigue response, damage initiation and progression during fatigue response, damage initiation and progression due to monotonic loading, and first matrix cracking and ultimate strength from a laminate stress-strain response. To perform these tasks the user must provide the appropriate output files from a MAC/GMC analysis. The MATLAB scripts described herein perform the various post processing tasks by parsing the output files from MAC/GMC analyses. Reference 8 describes a specific post processing task related to visualization of local stress and strain response within an RUC and may be consulted in conjunction with the material described.

4.1 Extraction of RUC Stiffness Related Properties.

The effective properties of a typical composite material can be extracted from the generic MAC/GMC output file (which has the extension ".out") by using the script "**ExtractRucStiffnesses**". There are nine effective elastic properties that are extracted from the output file by executing the command:

```
[A] =  ExtractRucStiffnesses (Title,FileId);
```

Where '**Title**' should be defined by the user depending on the analysis model. If the analysis is conducted by using the 'GMC' model then use:

```
Title='Effective Engineering Moduli';
```

If the analysis model is 'HFGMC' then use:

```
Title='EFFECTIVE MODULI';
```

'**FileId**' is the identification number of the output file, which is obtained by using:

```
FileId = fopen('filename.out','r');
```

where filename.out is the MAC/GMC output file. The material properties are stored in the variable **A** which contains a total of nine properties as defined below:

```
A(1) = E11S = 0.8895E+05
A(2) = N12S = 0.3639
A(3) = N13S = 0.3639
A(4) = E22S = 0.4470E+04
A(5) = N23S = 0.5093
A(6) = E33S = 0.4470E+04
A(7) = G23S = 0.1442E+04
A(8) = G13S = 0.1716E+04
A(9) = G12S = 0.1716E+04
```

Here, E denotes Young's modulus, N denotes Poisson's ratio and G denotes shear modulus in the various directions associated with an orthotropic material. It should be noted that although the effective properties are defined exactly the same way for both GMC and HFGMC analysis, at the time of this writing the title under which these are written to the MAC/GMC output file is different for GMC and HFGMC models. In future versions of MAC/GMC this will be corrected when the title would be the same for both types of analysis.

4.2 Extraction of Laminate Stiffnesses

The effective properties of a typical composite laminate can be extracted from the MAC/GMC output file by using the script "**ExtractLaminateStiffnesses**". There are four stiffness related properties that are extracted out of output by executing the command:

```
[Alam] = ExtractLaminateStiffnesses (Title,FileId);
```

where '**Title**' should be defined by the user by using the following command:

```
Title='Laminate Engineering Constants';
```

'**FileId**' is the identification number of the output file which is obtained by using:

```
FileId = fopen('filename.out','r');
```

where filename.out is the MAC/GMC output file. The laminate properties are stored in the variable **Alam** which contains a total of four effective in-plane laminate properties as defined below:

```
Alam(1) = Exx =  2.992E+05
Alam(2) = Nxy =  2.101E-01
Alam(3) = Eyy =  1.644E+05
Alam(4) = Gxy =  6.824E+04
```

Here, E denotes Young's modulus, N denotes Poisson's ratio and G denotes the shear modulus in the various directions associated with an orthotropic material.

4.3 Extraction of Number of Cycles to Failure From MAC/GMC Fatigue Simulation Output

A typical MAC/GMC fatigue analysis produces "**.out**" and "**_dam.out**" files, among other files, depending on the user-specified output requests. Both of the files mentioned above contain information pertaining to fatigue simulations, including damage level, number of cycles to failure, and the execution time. The function "**Ncycle**" takes the "**.out**" file as input and searches for a title "**TOTAL NUMBER OF CYCLES**" (not case sensitive) to extract the predicted number of cycles to failure. Additionally, upon user request, the function searches and extracts the execution time. In order to extract the information, the user should define:

```
Title='Total Number of Cycles';
```

And create a file id for output using:

```
outId=fopen('filename.out','r');
```

where **filename.out** is the MAC/GMC output file. The number of cycles to failure, **N**, and the execution time in seconds, **CPU**, are extracted from the output file by executing the command

```
[N,CPU] = Ncycle(Title,outid)
```

where **N** is the number of cycles to failure and **CPU** is the execution time in seconds. The title string above is not case sensitive. It should be noted that the function calls another function "extract_numbers" which is also listed in the Appendix along with Ncyle.m. The function "extract_numbers" helps parsing a single line from output for numbers. For e.g., if the line contains hours, minutes and secs for the execution time, then the function outputs these into a variable "r". The program computes the execution time using:

```
CPU  =  r(1)*3600  + r(2)*60 + r(3)
```

If CPU time is not required than user may specify

```
[N,~] = Ncycle(Title,outid) or simply

[N] = Ncycle(Title,outid)
```

Both are valid MATLAB commands when only the Number of cycles will be output.

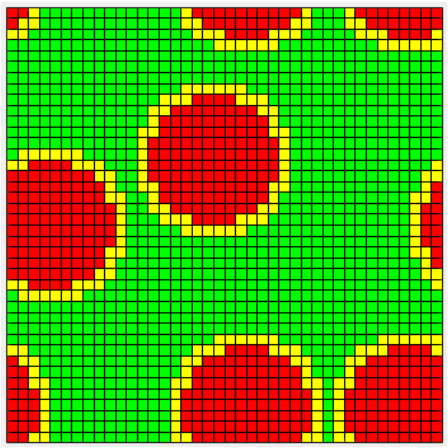
4.4 Extraction of First ply Matrix Cracking Strength or PLS

When modeling the stress-strain response of CMCs, it is of engineering interest to determine the first ply matrix cracking strength (proportional limit stress) and the ultimate strength. In order to obtain these quantities, the user performs a quasi-static loading analysis with incremental loading, and the output is stored as two columns representing stress and strain. The function "**PlsAndUS**" computes from this data the first matrix cracking strength and the ultimate strength. The syntax for the command is:

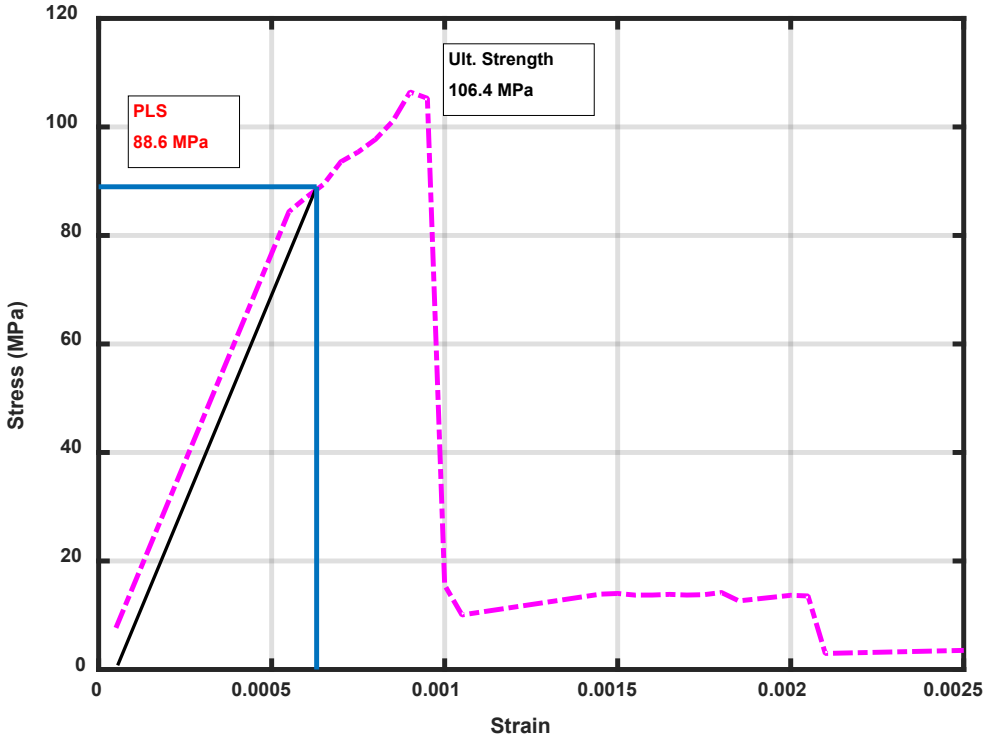
```
[Pls,Us] = PlsAndUs(Fname,E,offset,stp)
```

Where "**Pls**" is the first matrix cracking strength and "**Us**" is the ultimate strength for the composite. "**Fname**" is the name of output file containing the stress strain response information, **E** is initial Young's modulus of composite, "**offset**" is the strain offset, which is usually taken as 0.005 percent, and "**stp**" is the step size to perform the search for the intersection of the stress-strain curve with a straight line drawn through point [0,offset] with a slope of initial tangential modulus, E. As shown in Figure 12, the intersection of this straight line with the stress-strain curve defines the Pls. The stepsize "**stp**" is usually taken as $1/10^{th}$ of the offset. However, the "offset" and "stp" can be chosen according to the user's specific requirements. The Us is the absolute maximum of the stress-strain curve.

The **Pls** and **US** extraction for a typical transversely loaded CMC material is depicted in Figure 12(a) and (b). The RUC is made of four randomly distributed fibers (Fig. 12(a)). The fiber volume ratio (fvr) is 0.28, the matrix volume ratio (mvr) is 0.59 and the interface volume ratio (ivr) is 0.13. The extracted Pls for this composite is 88.6 MPa and the ultimate strength is 106.4 MPa. The initial transverse Young's modulus is 153.4 GPa.



(a)



(b)

Figure 12.—(a) Four fiber randomly arranged repeating unit cell (RUC) used in obtaining the transverse stress-strain response of a CMC composite system with fvr = 0.28, mvr = 0.59, ivr = 0.13. Red subcells are fibers, yellow subcells are interface and green subcells are matrix. (b) Transverse stress-strain response of a typical CMC composite system with, fvr = 0.28, mvr = 0.59, ivr = 0.13.

4.5 Plotting of the Local Stress and Strain Fields

MAC/GMC allows users to examine the local stress and strain fields in an RUC by providing detailed stress and strain output in each of the subcells for any load increment. The ***MATLAB** keyword (Refs. 2 and 3) allows users to specify for which time steps in the loading profile the local stress and strain fields are output. MAC/GMC outputs the following files, where **BaseFileName** is the base MAC/GMC file name without the extension '**.mac**'.

1. **BaseFileName_eps.dat**: Local strains
2. **BaseFileName_epsp.dat**: Local plastic strains and material Id information
3. **BaseFileName_epst.dat**: Local thermal strains
4. **BaseFileName_sig.dat**: Local stresses
5. **BaseFileName_x2.dat**: Subcell x_2-direction sizes
6. **BaseFileName_x3.dat**: Subcell x_3-direction sizes

For example, if the base file name is "Annika_2", then the MAC input file name is "Annika_2.mac". The output for the local stress information is in the file "Annika_2_sig.dat". The reader may look into References 2 and 3 for a more detailed description of these input and output files and their contents. The local stress response is extracted by executing the command:

```
[sig11,sig22,sig33,sig23,sig13,sig12,sigmean,sigeff,X,Y,RUC] =...
    ExtractStresses(BaseFileName)
```

Where the variables have the usual meaning as described below:

sig11 = σ_{11}; **sig22** = σ_{22}; **sig33** = σ_{33}; **sig23** = σ_{23}

sig13 = σ_{13}; **sig12** = σ_{12};

sigmean = $\left(\sigma_{11}+\sigma_{22}+\sigma_{33}\right)/3$

sigeff = effective or von Mises stress;

X and Y are the coordinate vectors of the RUC subcell grid. These contain the locations of the subcell boundaries in the two coordinate direction. **RUC** is the repeating unit cell matrix with material ids for each subcell.

The Local stress contours may be shown graphically by executing the MATLAB command:

```
Plot_Contours(Type,X,Y,Data,Title)
```

Where "**Data**" is any of the eight stresses described above. **Type** takes two values: '2-D' for two-dimensional or planar contours and '3-D' for three-dimensional contours. **Title** is the text that will be displayed as the plot title (including MATLAB text formatting). As an illustration refer to Figure 13, which can be produced by executing the command

```
Plot_Contours('2-D',X,Y,sig22,'\sigma_{22}')
```

The stress levels are shown as 2-D contours here. A colorbar is also provided to indicate the stress levels and the associated color code.

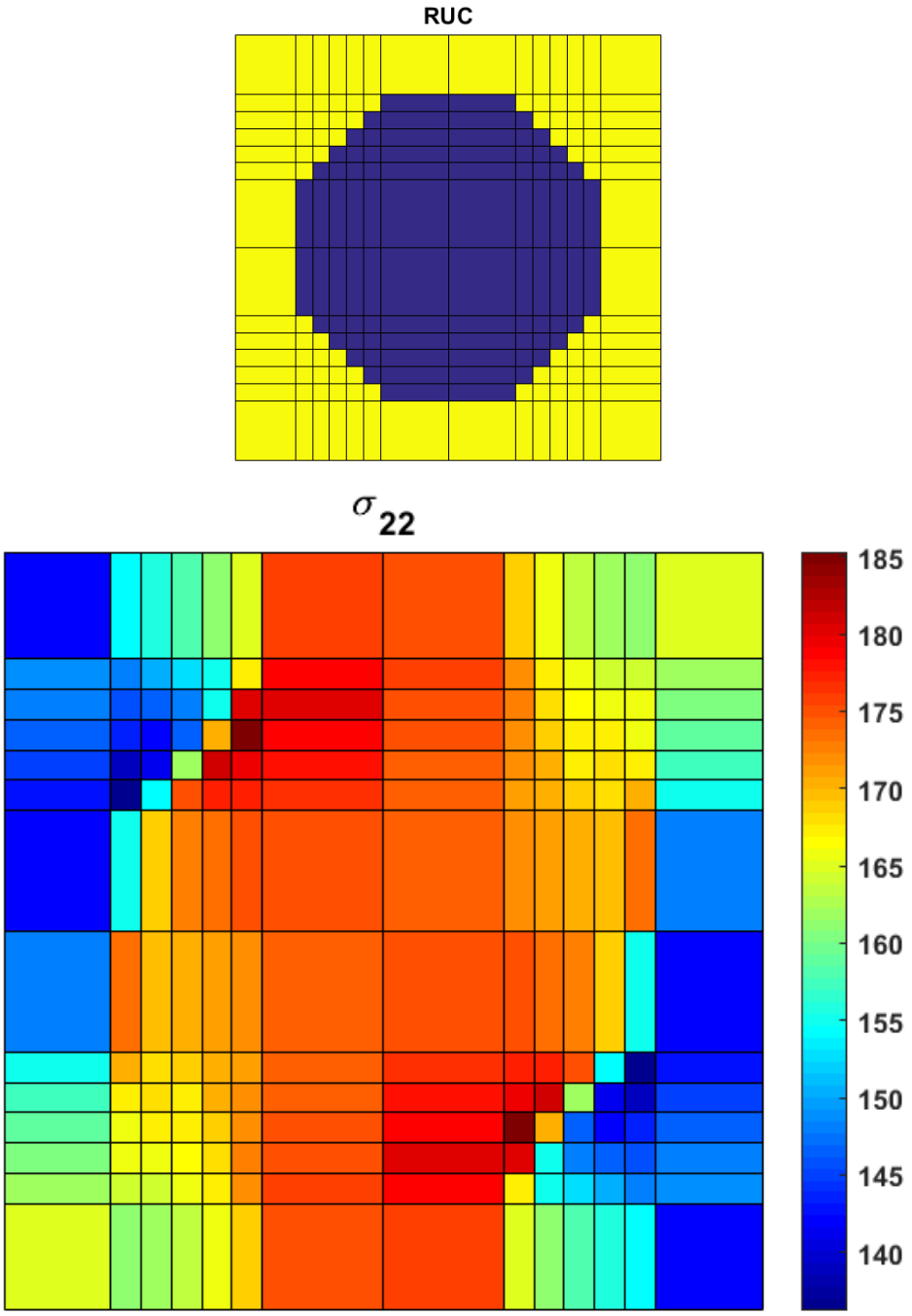


Figure 13.—Transverse stress σ_{22} contours for a PMC composite system with, fvr = 0.28, mvr = 0.59. Load is in 11 direction.

4.5.1 Local Strain Response

The local strain response can be obtained similarly by executing the command:

```
[Eps11,Eps22,Eps33,Eps23,Eps13,Eps12,X,Y] =...
    ExtractStrains(BaseFileName)
```

Where the variables have the usual meaning as described below:

Eps11 = ε_{11}; **Eps22** = ε_{22}; **Eps33** = ε_{33}; **Eps23** = γ_{23};

Eps13 = γ_{13}; **Eps12** = γ_{12};

It should be noted that the shear strains output are the engineering shear strains.

As an example, 3-D ε_{22} strain contours are shown in Figure 14. This figure is produced by executing the command:

```
Plot_Contours('3-D',X,Y,Eps22,'\epsilon_{22}')
```

The strain levels are shown as a surface over the plane x2-x3 in Figure 12.

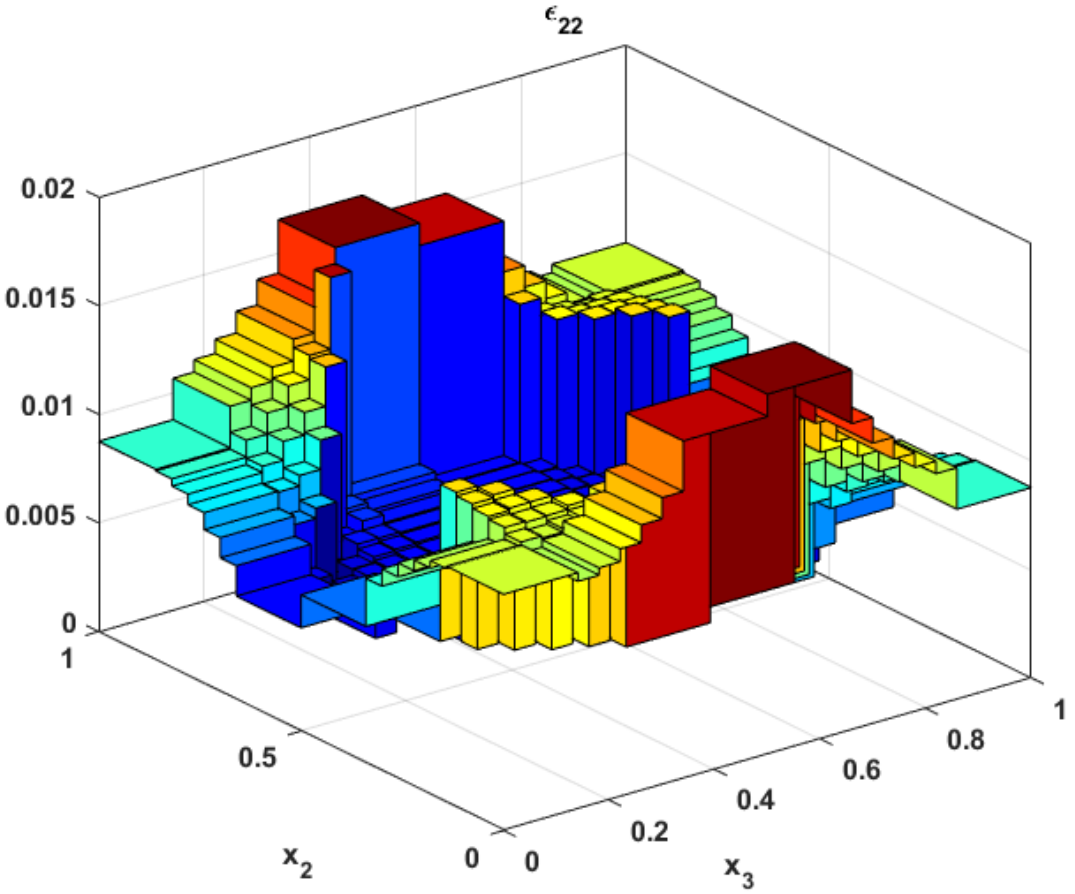


Figure 14.—Transverse strain ε_{22} contours for a PMC composite system with, fvr = 0.28, mvr = 0.59. Load is in 11 direction.

5.0 Miscellaneous MATLAB Recipes for MAC/GMC Runs

This section describes the use of scripts to perform MAC/GMC analyses that involve a combination of tools mentioned in this document. One example is a case where 100s of Monte Carlo simulations need to be run where each simulation may have a randomly chosen RUC and the task is to compute the number of cycles of load in fatigue to failure. Another example is to automatically run MAC/GMC several times to estimate a load versus number of cycles (SN) curve. Such analysis tasks typically involve manipulation of a MAC/GMC input file by changing the '*RUC' block of MAC/GMC input or the '*MECH' load block defining a single loading cycle, etc. The examples provided here are aimed at familiarizing the user with how to write the MATLAB scripts in general so that the user may be able to develop his or her own scripts for other purposes utilizing the ideas provided herein. These are not exhaustive as there can be any number of permutations and combinations of analyses and it is not possible to address them all here.

5.1 Monte Carlo Simulations

Assessing microstructure variability and how it affects typical composite properties and response is of considerable interest. In order to perform these tasks it is necessary to run a typical MAC/GMC problem repeatedly 100s of times while varying the definition of the RUC randomly on the fly. For example, if we want to assess the impact of property variability on the material response, the following flow chart of operations are needed as shown in Figure 15.

A typical MAC/GMC input file that is altered during each Monte Carlo simulation is shown in Figure 16. Note that for each run the *RUC block of the input (highlighted yellow in Fig. 16) is replaced by a randomly generated RUC representation and stored. One such *RUC block of data is shown in Figure 17.

Both the reference file "RefFile-RUC.MAC" and The *RUC block file "RUC8.txt" are provided as inputs to the function "ModifyRefFile," which creates a suitable input file to be run by MAC/GMC. The following lines of the script do this job:

```
Fname='RefFile-RUC.MAC' ; % Reference input file that gets modified
                          % for each Monte Carlo simulation
```

where Fname is the file name variable that contains the reference file data.

```
CL1=[18:57];     % lines in Ref file to be replaced on the fly for RUC
NSimuls = 100;   % Number of Monte - Carlo Simulations
```

where the CL1 variable contains the line numbers for which the data is to be replaced by the *RUC block data for each simulation.

```
for i  = 1: NSimuls
 RUCFilName=['RUC',num2str(i),'.txt']; % File containing *RUC block
                                       % details
 [A,Pls,Us]= ModifyRefFile(Fname,RUCFilName,CL1,i);
```

The above MATLAB script lines creates a *RUC block data file, "RUCFileName.txt?????" and calls the function ModifyRefFile to compute the stiffness, first matrix cracking strength and the ultimate strength of the composite material. This is performed "NSimuls" times, which for this case is 100.

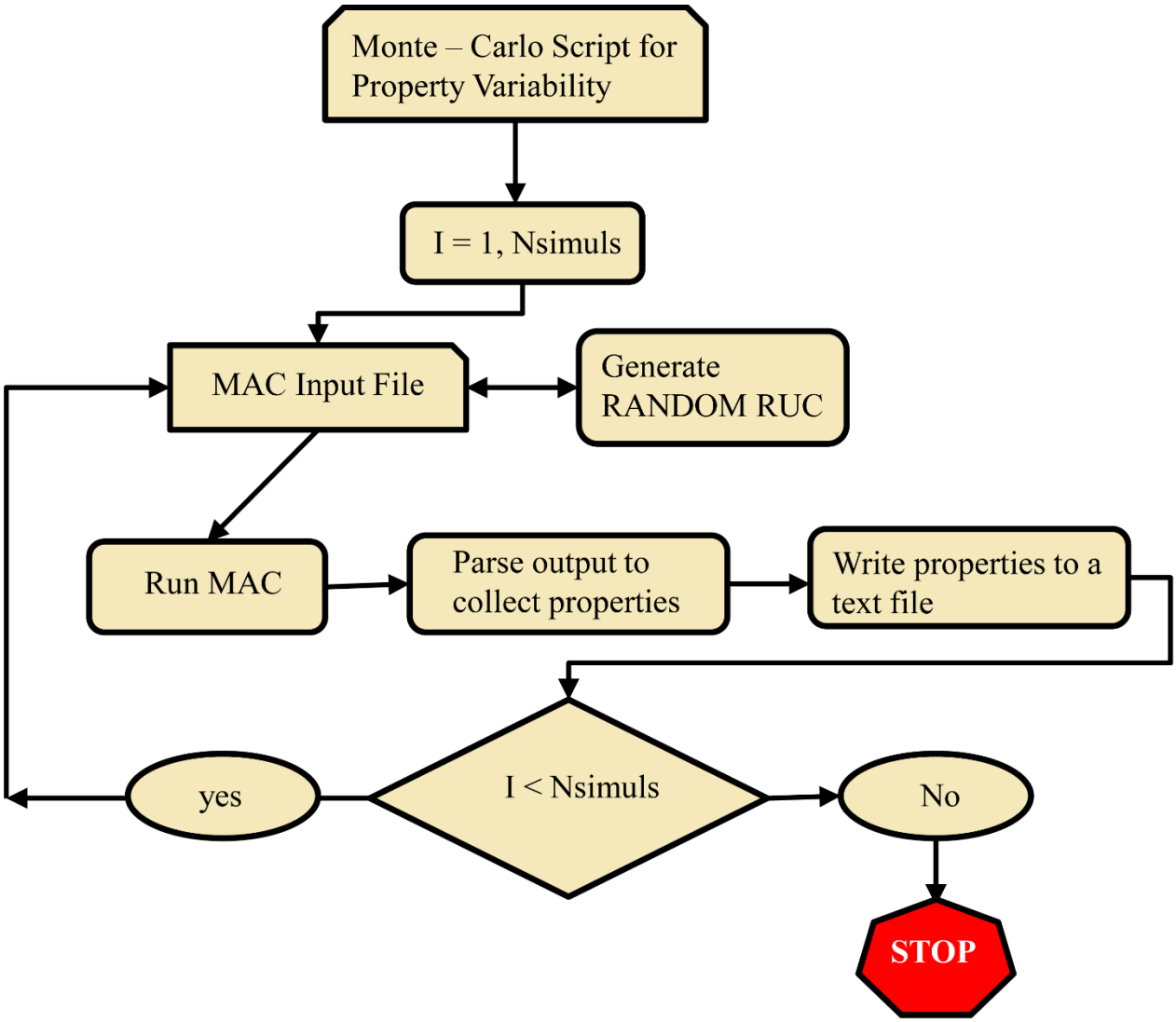


Figure 15.—Flow chart showing the MATLAB script for assessing microstructural variability of a typical composite system on properties.

```
RUC Basaed CMC Material Analysis- Reference File "RefFile-RUC.MAC"
*CONSTITUENTS
NMATS=3
# -- Hi-Nic Type S Fiber
M=1 CMOD=6 MATID=U MATDB=1 &
EL=385.0E3,385.0E3,0.17,0.17,164.5E3,5.2E-6,5.2E-6
# -- Siliconized SiC Matrix
M=2 CMOD=6 MATID=U MATDB=1 &
EL=327.0E3,327.0E3,0.22,0.22,134.0E3,5.4E-6, 5.4E-6
# -- BN coating
M=3 CMOD=6 MATID=U MATDB=1 &
EL=10.0E3,10.0E3,0.23,0.23,4.07E3,5.5E-6,5.5E-6
*RUC
MOD=2 ARCHID=99
NB=40 NG=40
H=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
L=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
SM=1,1,1,1,1,1,1,1,1,1,1,3,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,3,1

. . . [repeated 40 times] ...
*MECH
   LOP=2
   NPT=2 TI=0.,100.0 MAG=0.,0.005 MODE=1
*SOLVER
   METHOD=1 NPT=2 TI=0.,100.0 STP=1.0 ITMAX=250 ERR=1.E-3
    NLEG=3 NINTEG=5
*FAILURE_SUBCELL
   NMAT=3
   MAT=1 NCRIT=1
   CRIT=1 X11=1.8E3 X22=1.8E3 X33=1.8E3 X23=0.6E3 X13=0.6E3 &
    X12=0.6E3 COMPR=OFF ACTION=1
   MAT=2 NCRIT=1
   CRIT=1 X11=600.0 X22=600.0 X33=600.0 X23=350.0 X13=350.0 &
    X12=350.0 COMPR=OFF ACTION=1
#--    CMOD=4 PDMOD=99
#--    K11=-1.180E5 K22=-1.18E3
   MAT=3 NCRIT=1
   CRIT=1 X11=70.0 X22=70.0 X33=70.0 X23=45.0 X13=45.0 &
    X12=45.0 COMPR=OFF ACTION=1
*FAILURE_CELL
NCRIT=1
CRIT=2 X11=0.002 X22=0.002  X33=0.05 X23=0.05 X13=0.05 X12=0.05 &
COMPR=SAM ACTION=-1
*PRINT
  NPL=6
*XYPLOT
  FREQ=1
  MACRO=1
  NAME=Stress-Strain X=2 Y=8
  MICRO=0
*END
```

Figure 16.—Typical *RUC block to be altered and replaced within a MAC/GMC input file during Monte Carlo simulations.

```
*RUC
MOD=2 ARCHID=99
NB=40 NG=40
H=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1
L=1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1
SM=1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,3,2,2,3,1,1,
1,1,1
SM=1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,3,3,1,
1,1,1
SM=1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2,2,2,3,1,
1,1,1
SM=1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2,2,2,3,3,
1,1,1
SM=1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,
3,3,1
SM=3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,3,3
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,
3,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,
3,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,
3,3,2
SM=2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,
1,3,2
SM=2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,
1,3,2
SM=2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,
1,3,2
SM=2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,
1,3,2
SM=2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,
3,3,2
SM=2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,
3,2,2
SM=2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,3,
3,2,2
SM=2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,3,3,3,1,1,1,1,3,3,3,
2,2,2
SM=2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,2,2,
2,2,2
SM=2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
```

```
SM=2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,3,3,1,1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,3,3,3,1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,2,2,2,2,2,2,2,
2,2,2
SM=2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,1,1,1,3,3,3,2,2,2,
2,2,2
SM=3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,3,3,2,2,2,
2,3,3
SM=1,1,1,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,3,2,2,2,
3,3,1
SM=1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,
1,1,1
SM=1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,3,2,2,
1,1,1
SM=1,1,1,1,1,1,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,3,2,3,
1,1,1
SM=1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,3,2,3,
1,1,1
SM=1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,3,1,1,1,1,1,1,1,1,1,1,1,3,2,3,
1,1,1
SM=1,1,1,1,1,1,1,3,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,
3,1,1,1
```

Figure 17.—Typical *RUC block to be inserted into a MAC input file during Monte Carlo simualtions.

The two main script files that perform the Monte Carlo simulation for property variability studies are "MonteCarloScript_PropertyVariability.m" and "ModifyRefFile.m". These are given in full in the Appendix: MATLAB Scripts. The results for all 100 runs are stored in a text file "DBase.txt" that can be used later for post processing either in MATLAB or EXCEL.

Other tools that are called within the program are "PlsAndUs.m" and "ExtractRucStiffnesses.m." These extract the strengths and stiffnessess, respectively, and have been described previously in Sections 4.4 and 4.1.

With slight modification to the above scripts, one could perform Monte Carlo simulations for a laminate. It should be noted that "ExtractLaminateStiffnesses.m" will have to be used to extract stiffnesses in this case. Furthermore, since there will be several layers, for each layer the "*RUC block" needs to be modified for each simulation.

The above procedure can also be utilized for running Monte Carlo simulations to assess the fatigue strength variability. Here one will have to provide the appropriate load block information for fatigue and the failure criteria with in the reference file.

5.2 Fatigue Strength vs. Life: SN Curve

The entire SN Curve that is typical for visualizing the fatigue response of materials involves repeated runs of MAC/GMC. The load block information is usually provided under the *MECH keyword block where the fatigue loading block, which typically consists of the … min – max – min loads, is provided. This operation is performed by calling the function "SNFatigueCurves.m" with the syntax:

```
[N] = SNFatigueCurves(InF,S,Prog)
```

Where "InF" the name of MAC/GMC input file. The " *MECH load block" within the MAC/GMC input file is altered for each block of fatigue loading. The variable "S" contains the load levels. The variable "Prog" defines the MAC/GMC executable. For example, the input file "NewFatGMC.MAC," which represents a $[+/- 70]_s$ laminate, is shown in Figure 18. The load levels and executable are defined by the following script:

```
S  = [70:-5:20];
Prog= 'mac4z-3_2';
```

Figure 19 shows the generated SN curve for this case. Once the MAC/GMC analysis is completed, the output is scanned for the fatigue life information using the function "Ncycle" which was explained previously in Postprocessing Scripts Section 4.3.

```
Fatigue Damage Analysis Unidirectional lamina (Or. RUC)
*CONSTITUENTS
NMATS=2
# -- Graphite fiber
M=1 CMOD=6 MATID=U MATDB=1 &
EL=388.2e3,7.6E3,0.41,0.45,14.9E3,-0.68E-6,9.74e-6
# -- Epoxy matrix
M=2 CMOD=6 MATID=U MATDB=1 &
EL=3.45e3,3.45e3,0.35,0.35,1.278E3,45.E-6,45.E-6
*LAMINATE
NLY=4
LY=1 THK=0.25 ANG=70 MOD=2 ARCHID=1 VF=0.6 F=1 M=2
LY=2 THK=0.25 ANG=-70 MOD=2 ARCHID=1 VF=0.6 F=1 M=2
LY=3 THK=0.25 ANG=-70 MOD=2 ARCHID=1 VF=0.6 F=1 M=2
LY=4 THK=0.25 ANG=70 MOD=2 ARCHID=1 VF=0.6 F=1 M=2
*MECH
LOP=1
NPT=4 TI=0.,50.,150.,200. MAG=0,SL.,-SL.,0 MODE=2,2,2
*SOLVER
METHOD=1 NPT=4 TI=0.,50,150.,200. STP=10.,10.,10.
NLEG=1 NINTEG=1
*DAMAGE
MAXNB=100 DINC=0.2 DMAX=0.9999 BLOCK=0.,200.
NDMAT=2
MAT=1 MOD=2 SU1=3500.,350.0,350.0,130.0,250.0,250.0 &
SU2=2000.,200.0,200.0,75.0,143.0,143.0 &
N1=1000,1000,1000,1000,1000,1000 &
N2=300000000,300000000,300000000,300000000,300000000,300000000
MAT=2 MOD=1 ANG=0. BN=0. BP=0. OMU=1. OMFL=1. OMM=1. ETU=1. &
ETFL=1. ETM=1. BE=9. A=0.05 SFL=27. XML=150. SU=80.
*FAILURE_SUBCELL
NMAT=2
MAT=1 NCRIT=1
CRIT=1 X11=3500. X22=350.0 X33=350.0 X23=130.0 X13=250.0 X12=250.0 &
COMPR=SAM
MAT=2 NCRIT=1
CRIT=1 X11=80. X22=80. X33=80. X23=40. X13=40. X12=40. &
COMPR=SAM
*FAILURE_CELL
NCRIT=1
CRIT=2 X11=0.05 X22=0.05 X33=0.05 X23=0.05 X13=0.05 X12=0.05 &
COMPR=SAM
*PRINT
NPL=3
*END
```

Figure 18.—A typical MAC input for fatigue strength analysis.

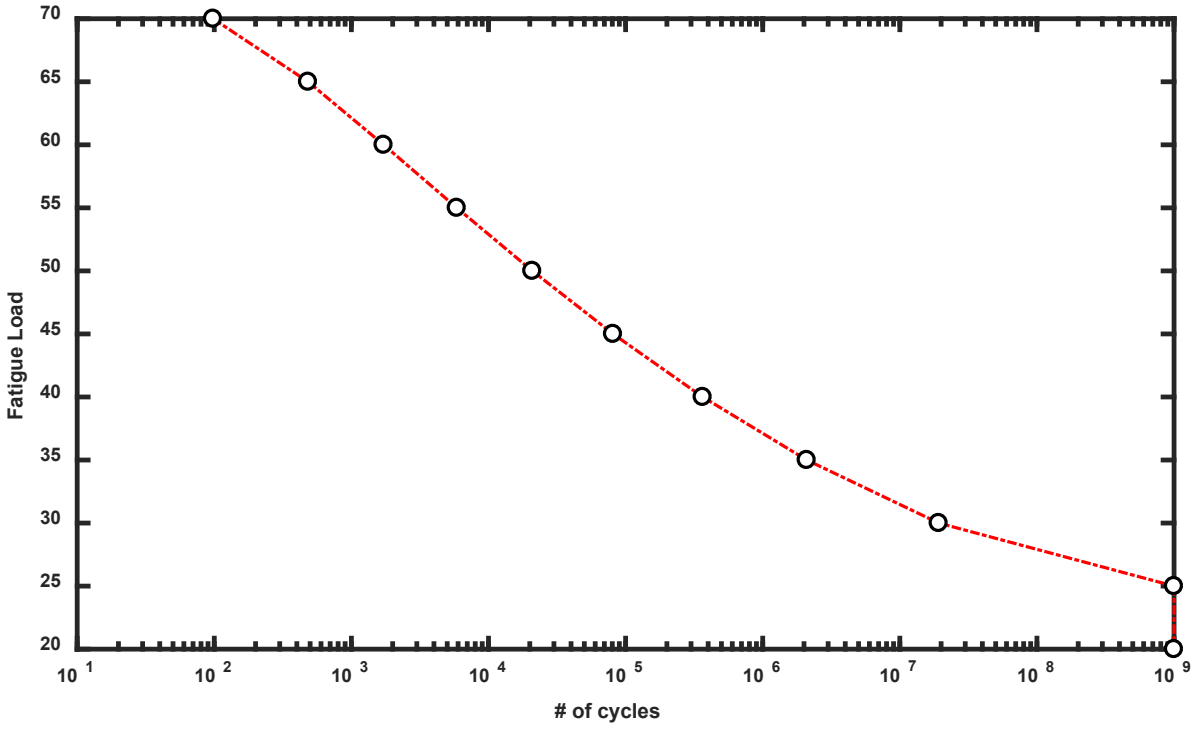


Figure 19.—A typical SN curve for a $[+/- 70°]_S$ Laminate.

5.3 Generation of Random RUCs for Use in Monte Carlo Simulations

It is often useful to generate RUCs with randomly distributed fibers and store them for subsequent investigation of the effect of microstructure variability on strength, stiffness and other response quantities. The following scripts provide a means to generate the *RUC block of MAC/GMC input and to store them in an orderly manner. The scripts "CMC_RUCsForMCRuns" and "CMC_RUCs_withsqFibers_MCRuns" (Appendix) create a prescribed number of circular and square fiber CMC RUCs by calling repeatedly the functions described earlier under the preprocessing scripts CM_Ruc and RechFiberPlacementWithMirroring.

For CMC RUCs with circular fibers, we utilize the CMC_Ruc function. The inputs to this function are described below:

```
Radf=6;           % Fiber Radius
Radi=7.2;         % Interface Radius
nx=40;ny=40;      % Number of subcells in each direction
Nfibers=4;        % Number of fibers
nrep=1;           % Nuber of reps for tiling. Not used at all here.
NSimuls = 100;    % Number of MAC runs
Fn='RUC';         % Generic prefix for file name
```

The call the function CM_Ruc, the following syntax is used:

```
[FVR,IVR,MVR,RUC,~,~,~] = CMC_Ruc ( Radf,Radi,nx, ny, Nfibers,nrep)
```

Once the RUC is generated, it is written to a file by calling the function

```
WriteSmallRuc(RUC,Fname)
```

The files are named as "RUCnn.txt" where nn is a number between 1 thru the total number of RUCs requested to be generated. The entire script is provided in the Appendix.

Similar to the above, in order to generate a randomly distributed square fiber RUCs we utilize the function "RectFiberPlacementWithMirroring." The following inputs are needed to call the function:

```
fvr     = .28;    % Fiber Volume Ratio
ivr     = 0.13;   % Interface Volume Ratio
Nfibers=4;        % Number of fibers
nrep=1;           % Nuber of reps for tiling. Not used at all here.
Nsimuls = 100;    % Number of MC runs
Fn='RUCSq';       % Generic prefix for file name
Eclr=0.01;        % Interfiber clearance.
```

The call to the function has the syntax:

```
[RUC,L,H,success] = RectFiberPlacementWithMirroring(Nfibers,fvr,ivr,Eclr)
```

The value for the variable "success" takes either "0" or "1". If the program is able to place all the fibers successfully then a value of "1" is returned otherwise a warning message is displayed for the user to make appropriate changes and try again. Once the RUC is generated it is written to a file by calling the function

```
WriteRandSqRuc (RUC,Fname,H,L)
```

The files are named as "RUCsqnn.txt" where nn is a number between 1thru the total number of RUCs requested to be generated. The entire script is provided in the Appendix.

6.0 Concluding Remarks

In this document we have compiled a number of scripts and functions written in the MATLAB programming language that are useful in automating various MAC/GMC analyses tasks. Illustrative examples that show the usage of the functions and scripts are provided throughout the document in order to help new users perform MAC/GMC analyses tasks with ease as well as help them develop new analyses scripts. As mentioned before, the tools are not meant to be exhaustive, but rather to provide a flavor of what can be done and automated. They provide enough information for developing other similar scripts as needed.

Appendix—MATLAB Scripts

This section includes all of the source code for the MATLAB scripts that have been described herein. These scripts can be copied and pasted into MATLAB and can be executed. It is assumed here that the user has some basic knowledge of how to execute MATLAB scripts, their syntax, and some programming knowledge. Modifications may be made to the existing scripts in order to tailor the scripts to meet their individual requirements.

Preprocessing Scripts

All the MATLAB preprocessing scripts for MAC/GMC analyses described in Section 2 are provided below:

Plot a Repeating Unit Cell

This function takes the basic RUC definition and shows it pictorially with different colors for each constituent fiber, matrix and interface. This function only works for uniform grid with square size subcells.

```
function [] = plotRUC (RUC)
% Developed by Pappu L.N. Murthy
%    Plot RUC
  xrange = [1.5 size(RUC,2)+0.5];
  yrange = [1.5 size(RUC,1)+0.5];
imagesc(xrange,yrange,RUC);
% Find Unique elements in RUC
U=unique(RUC);nmat=length(U);
cmap=[1 0 0];
% setup Color scheme
if (nmat==2)
cmap=[1 0 0;1 1 0];
elseif(nmat==3)
cmap=[1 0 0; 0 1 0; 1 1 0];
end
set(gcf,'colormap',cmap)
grid on;
set (gca,'GridColor','k','LineWidth',.25,'GridAlpha',1);
axis image;
set(gca,'XTick',1:size(RUC,2)+1,'YTick',1:size(RUC,1)+1, ...
    'XTickLabel','','YTickLabel','');
```

Plot a Repeating Unit Cell With Nonuniform Subcells

This function takes as input the basic RUC definition, the H and L vectors defining the subcell discretization in the height and width directions of the RUC and shows it pictorially with different colors for each constituent fiber, matrix and interface. This function can take both uniform and nonuniform subcells within the RUC.

```
function []= showRuc(RUC,H,L)
% ------------------------------------------------------------------------
-----
% This program is for square RUC's with Square Fiber shapes only
% Developed by Pappu L.N. Murthy
% MultiScale and MultiPhysics Modeling Branch
% Date: Oct, 3, 2016.
% ------------------------------------------------------------------------
-----
[m,n]=size(RUC); Nmat = length(unique(RUC));
px=0;py=0;
switch Nmat
    case 1
        cmap=[0 1 1];
    case 2
        cmap=[0 1 1;0 1 0];
    case 3
        cmap= [0 1 1;0 1 0;1 1 0];
    case 4
%         cmap= [0 1 1;0 1 0;1 1 0;1 0 0];
      cmap= [1 0 0;0 1 0;1 1 0;1 1 1];
end
for i = 1:m
    for j=1:n
        rectangle('Position',[px py L(j) H(i)],'FaceColor',...
        cmap(RUC( i ,j),:));
    px=px+L (j);
    end
    px=0;
    py=py+H (i);
end
 xlim([0 sum(L )]);ylim([0 sum(H )]);
 daspect([1,1,1]);
 axis off;
```

Extraction of RUC and H and L Information From a "*RUC" Block of a Typical MAC/GMC Input File

This function takes as input a typical input section of MAC input block represented by '*RUC' key word. It outputs the definitions for RUC, and the height and width direction subcell size information in H and L vectors.

```
function[RUC,H,L] = ExtractSqRuc(f,Ln)
% This function Extracts the info regarding RUC and the subcell
% dimenstions in H and L from *RUC block of MAC input data. Note that
% there should not be any blanks in the input and the routine is for
% square RUC (NB = NG) or rectangular where (NB ≠ NG);
% Developed by Pappu L.N. Murthy, Date: June 2, 2016,
% Multiphysics and Multiscale Modeling Branch
% NASA Glenn Research Center

if (nargin==1)
    Ln=6; % Note for *RUC block this number is 6 However, for
          % *LAMINATE Block this is line #7
end
fid1=fopen(f,'r');
C1 = textscan(fid1,'%s','Delimiter','\n');
C1 = C1{1};
fclose(fid1);
format_dims=('NB=%d NG=%d');
format_strH='H=%f';
format_strL='L=%f';
StartLine=Ln-3;
n=cell2mat( textscan(C1{StartLine},format_dims));
format_strH = [format_strH, repmat(',%f',1,n(1)-1)];
format_strL = [format_strL, repmat(',%f',1,n(2)-1)];
C=C1(Ln:end);
H=cell2mat(textscan(C1{StartLine+1},format_strH));
L=cell2mat(textscan(C1{StartLine+2},format_strL));
% RUC=[];
RUC=zeros(n);
for i=1:length(C)
 s=textscan(C{i},'%s','Delimiter','=');
 a=s(1);
 b= a{1};
 f=textscan(b{2},'%s','Delimiter',',');
 g=cell2mat(f{1});
 RUC(i,:)=str2num(g)';
end
RUC=flipud(RUC);
```

Write *RUC block for a Typical MAC/GMC Input File

This function based on a given RUC, H and L definitions, generates the input section of '*RUC' block for a typical MAC input data set. The output is a text file and can be readily inserted into a MAC input dataset.

```
function [] = WriteRuc (RUC,FileIn,H,L)
% This function enables writing of a typical *RUC block % of MAC input
data.
% Developed by Pappu L.N. Murthy, Date: June 2, 2016,
% Multiphysics and Multiscale Modeling Branch
% NASA Glenn Research Center
[nx,ny]=size(RUC);
fidN=fopen(FileIn,'wt'); % MAC compatible input file is % written here
% Create the RUC Part of the Mac Input for any user %    % defined
archID, 99 .

fprintf(fidN,'%s\n', '*RUC','MOD=2 ARCHID=99');
fprintf(fidN,'NB=%2i NG=%2i\n',nx,ny);
% H to Mac input file.
format_str = 'H=%d';n=length(H);
if n  > 1
    format_str = [format_str, repmat(',%d', 1, n-1)];
end
format_str = [format_str, '\n'];
fprintf(fidN, format_str, H);
% Write L to Mac Input file
if (nx~=ny)
    format_str = 'L=%d';n=length(L);
    if n  > 1
        format_str = [format_str, repmat(',%d', 1, n-1)];
    end
    format_str = [format_str, '\n'];
else
    format_str(1)='L';
end
fprintf(fidN, format_str, L);
% Write each row of the cell
format_str='SM=%d';n=size(RUC,2);
if n>1
    format_str = [format_str, repmat(',%d',1,n-1)];
end
format_str = [format_str, '\n'];
RUC=flipud(RUC);
fprintf(fidN,format_str,RUC');
fclose(fidN);
```

Writing Very Large RUC Input Blocks Where MAC/GMC Continuations are Used

This function is useful for writing input blocks of '*RUC' where the RUC size is extremely large and the input requires continuation lines in the text file. MAC program considers '&' at the end of any input data line as a continuation.

```
function [] = WriteBigRuc (RUC,FileIn,H,L)
[nx,ny]=size(RUC); Nentry=200;
% Create the RUC bloc of Mac Input for any archID, 99 .
fidN=fopen(FileIn,'wt'); % MAC compatible input file is written here
fprintf(fidN,'%s\n', '*RUC','MOD=2 ARCHID=99');
fprintf(fidN,'NB=%2i NG=%2i\n',nx,ny);
% H to Mac input file.
format_str = 'H=%d';n=length(H);
if n > Nentry
    nlines=fix(n/Nentry);Rem=mod(n,200);
    format_str1  = [format_str,repmat(',%d',1,Nentry-1),', &\n'];
    format_str2 = ['%d',repmat(',%d',1,Nentry-1),',&\n'];
    for lines = 1:nlines
        if (lines == 1)
            fprintf(fidN, format_str1, ...
                H( (lines-1)*Nentry+1: Nentry*lines));
        else
            fprintf(fidN, format_str2, ...
                H( (lines-1)*Nentry+1: Nentry*lines));
        end
    end
    format_str3 = ['%d',repmat(',%d',1,Rem-2),',%d\n'];
            fprintf(fidN, format_str3, H(Nentry*lines+1:end));
end
% Write L to Mac Input file
 format_str = 'L=%d';n=length(L);
 if n  > Nentry
     nlines=fix(n/Nentry);Rem=mod(n,200);
     format_str1  = [format_str,repmat(',%d',1,Nentry-1),', &\n'];
     format_str2 = ['%d',repmat(',%d',1,Nentry-1),', &\n'];
     for lines = 1:nlines
         if (lines == 1)
             fprintf(fidN, format_str1, ...
                 L( (lines-1)*Nentry+1: Nentry*lines));
         else
             fprintf(fidN, format_str2, ...
                 L( (lines-1)*Nentry+1: Nentry*lines));
         end
     end
        format_str3 = ['%d',repmat(',%d',1,Rem-2),',%d\n'];
        fprintf(fidN, format_str3, L(Nentry*lines+1:end));
 end
% Write each row of the cell
RUC=flipud(RUC);
for Rows = 1:length(H)
    xx=RUC(Rows,:);
```

```
    format_str='SM=%d';n=length(xx);
    if n > Nentry
    nlines=fix(n/Nentry);Rem=mod(n,200);
    format_str1  = [format_str,repmat(',%d',1,Nentry-1),', &\n'];
    format_str2 = ['%d',repmat(',%d',1,Nentry-1),', &\n'];
    for lines = 1:nlines
        if (lines == 1)
            fprintf(fidN, format_str1,...
                xx( (lines-1)*Nentry+1: Nentry*lines));
        else
            fprintf(fidN, format_str2, ...
                xx( (lines-1)*Nentry+1: Nentry*lines));
        end
    end
    format_str3 = ['%d',repmat(',%d',1,Rem-2),',%d\n'];
            fprintf(fidN, format_str3, xx(Nentry*lines+1:end));
    end
end
fclose(fidN);
```

Square Packed Circular Fibers RUC Generation

This function generates an RUC for a CMC material given the fiber volume and the interface volume ratios as well as the number of subcells within the interface. The RUC is for a square packed configuration. If the interface volume ratio is given as '0', the program generates a PMC RUC in which case the user needs to specify the parameter Nw which is basically the number of subcells in width or height directions.

```
function[fvr,mvr,ivr,RUC ]=SquarePack(Fvr,Ivr,nrep,Ni,Nw)
% %
% % Program that generates a Square packing CMC Ruc with
% % interface.
% % Fiber radius and Interface thickness are user
% % definable.
% % Developed by Pappu L.N. Murthy, Date June, 30, 2015;
% %

M=2;F=1;I=3;
W=1;H=1;
TA = W*H;
FA = Fvr*TA;
Radf = sqrt(FA/pi);
IA = Ivr*TA;
Radi = sqrt((FA+IA)/pi);

flag=0;
if ( abs(Radf-Radi)>=eps  )
    flag=1;
end
if(flag)
    ti=Radi-Radf;
    Radi=round(Ni*Radi/ti);
    Radf=round(Ni*Radf/ti);
    ny=round(Ni*W/ti);
else
    ny=Nw;
    Radf= round( sqrt ( ((ny*ny)*Fvr)/pi));
    Radi=Radf;
end
if (mod(ny,2))
    ny=ny-1;
end

nx=ny;

Lbase=ones(1,ny);Hbase=ones(1,nx);
base = M*ones(nx ,ny );              % Square RUC
% Fiber volume ratio Optimization
Fact=(.95:0.001:1.2);
fv=zeros(1,length(Fact));
iv=fv;
Raf=fv;Rai=fv;
```

```
% define center of RUC
cx=nx/2;cy=ny/2;
for ii = 1:length(Fact)
    Rf=Fact(ii)*Radf;Ri=Fact(ii)*Radi;
    for i=1:nx
        for j=1:ny
            radx = (j-0.5);rady=(i-0.5);
            dist  = sqrt ( (cx-radx)^2+ (cy-rady)^2 );
            if (dist<=Rf )
                base(i,j)=F;
            elseif (dist>Rf && dist<=Ri)
                base(i,j)=I;
            end
        end
    end

    Ma= base ==M;Marea=sum(Ma(:));
    Fa= base ==F;Farea=sum(Fa(:));
    Ia= base ==I;Iarea=sum(Ia(:));
    Ta=Marea+Farea+Iarea;
    fvr= Farea/Ta; ivr=Iarea/Ta;
    fv(ii)=fvr;
    iv(ii)=ivr;
    Raf(ii)=Rf;Rai(ii)=Ri;
    if (fv(ii)>=Fvr)
        break
    end
end

Rf=Raf(ii);
if (ii>1)
    if ( abs(Fvr-fv(ii)) > abs(Fvr-fv(ii-1)) )
        Rf=Raf(ii-1);
    end
end
%-------------------------------------------------------------
if (flag)
    % Interphase Optimzation
    base(base(:)==I)=M;
    Fact=(Rf/Radi:0.001:1.2);
    fv=zeros(1,length(Fact));
    iv=fv;
    Rai=fv;
    for ii = 1:length(Fact)
        Ri=Fact(ii)*Radi;
        for i=1:nx
            for j=1:ny
                radx = (j-0.5);rady=(i-0.5);
                dist  = sqrt ( (cx-radx)^2+ (cy-rady)^2 );
                if (dist<=Rf )
                    base(i,j)=F;
                elseif (dist>Rf && dist<=Ri)
```

```
                    base(i,j)=I;
                end
            end
        end

        Ma= base ==M;Marea=sum(Ma(:));
        Fa= base ==F;Farea=sum(Fa(:));
        Ia= base ==I;Iarea=sum(Ia(:));
        Ta=Marea+Farea+Iarea;
        fvr= Farea/Ta;ivr=Iarea/Ta;
        fv(ii)=fvr;
        iv(ii)=ivr;
        Rai(ii)=Ri;
        if (iv(ii)>=Ivr)
            break
        end
    end
    Ri=Rai(ii);
    if (ii>1)
        if ( abs(Fvr-fv(ii)) > abs(Fvr-fv(ii-1)) )
            Ri=Rai(ii-1);
        end
    else
    end
%-------------------------------------------------------------
   % Generate current Ruc;
     base = M*ones(nx ,ny );             % Square RUC
         for i=1:nx
            for j=1:ny
                radx = (j-0.5);rady=(i-0.5);
                dist  = sqrt ( (cx-radx)^2+ (cy-rady)^2 );
                if (dist<=Rf )
                    base(i,j)=F;
                elseif (dist>Rf && dist<=Ri)
                    base(i,j)=I;
                end
            end
         end
    Ma= base ==M;Marea=sum(Ma(:));
    Fa= base ==F;Farea=sum(Fa(:));
    Ia= base ==I;Iarea=sum(Ia(:));
    Ta=Marea+Farea+Iarea;
    fvr= Farea/Ta; mvr=Marea/Ta; ivr=Iarea/Ta;
    RUC=base;
%-------------------------------------------------------------
else
%-------------------------------------------------------------
    % PMC. No interface.
base = M*ones(nx ,ny );             % Square RUC
    for i=1:nx
        for j=1:ny
            radx = (j-0.5);rady=(i-0.5);
```

```
            dist  = sqrt ( (cx-radx)^2+ (cy-rady)^2 );
            if (dist<=Rf )
                base(i,j)=F;
            end
        end
    end
        Ma= base ==M;Marea=sum(Ma(:));
        Fa= base ==F;Farea=sum(Fa(:));
        Ta=Marea+Farea;
        fvr= Farea/Ta; mvr=Marea/Ta; ivr=Iarea/Ta;
        RUC=base;
end
```

Hexagonally Packed Circular Fibers RUC Generation

This function generates an RUC for a hexagonal packing arrangement for the fibers for both CMC and PMC materials. Typical RUC in this case would include one fiber in the center and four quadrants of the fiber in each of the four corners of a rectangle.

```
function[fvr,mvr,ivr,RUC]= HexaPack (Fvr,Ivr,nrep,Ni,Nw)
% Program that generates a Hexagonal packing CMC Ruc with    %
interface.
% Fiber radius and Interface thickness are user difinable.
% Developed by Pappu L.N. Murthy,
% format long;clear all;
if (nargin==3)
    Ni=1;
end
M=2;F=1;I=3;

W=1;H=W/sqrt(3);
TA = W*H;
FA = Fvr*TA;
Radf = sqrt(FA/(2*pi));
IA = Ivr*TA;
Radi = sqrt((FA+IA)/(2*pi));
flag=0;

% Determine if fiber radius and interface radius are same
if ( abs(Radf-Radi)>=eps  )
    flag=1;
end
if(flag)
    ti=Radi-Radf;
    Radi=round(Ni*Radi/ti);
    Radf=round(Ni*Radf/ti);
    ny=round(Ni*W/ti);
else
    ny=Nw;
    Radf= round( sqrt ( ((ny*ny/sqrt(3))*Fvr)/(2*pi)));
    Radi=Radf;
end

if (mod(ny,2))
    ny=ny-1;
end

nx=fix(1.732\ny);
if(mod(nx,2))
    nx=nx-1;
end
Lbase=ones(1,ny);Hbase=ones(1,nx);
base = M*ones(nx/2,ny/2);
% Rectangular RUC with two of the opposite corners made of % fibers
with interface.
```

```
% Fiber volume ratio Optimization
Fact=(.95:0.001:1.2);
fv=zeros(1,length(Fact));
iv=fv;Raf=fv;Rai=fv;
for ii = 1:length(Fact)
    Rf=Fact(ii)*Radf;Ri=Fact(ii)*Radi;
    for i=1:nx/2
        for j=1:ny/2
            dist=sqrt(i^2+j^2);
            dist2=sqrt((nx/2-i+1)^2 + (ny/2-j+1)^2);
            if (dist<=Rf || dist2<=Rf)
                base(i,j)=F;
            elseif (dist>Rf & dist<=Ri || dist2>Rf &  dist2<=Ri)
                base(i,j)=I;
            end
        end
    end

    Ma= base==M;Marea=sum(Ma(:));
    Fa= base==F;Farea=sum(Fa(:));
    Ia= base==I;Iarea=sum(Ia(:));
    Ta=Marea+Farea+Iarea;
    fvr= Farea/Ta;ivr=Iarea/Ta;
    fv(ii)=fvr;
    iv(ii)=ivr;
    Raf(ii)=Rf;Rai(ii)=Ri;
    if (fv(ii)>=Fvr)
        break
    end
end
Rf=Raf(ii);
if (ii>1)
    if ( abs(Fvr-fv(ii)) > abs(Fvr-fv(ii-1)) )
        Rf=Raf(ii-1);
    end
end
if (flag)
    % Interphase Optimzation
    base(base(:)==I)=M;
    Fact=(Rf/Radi:0.001:1.2);
    for ii = 1:length(Fact)
        Ri=Fact(ii)*Radi;
        for i=1:nx/2
            for j=1:ny/2
                dist=sqrt(i^2+j^2);
                dist2=sqrt((nx/2-i+1)^2 + (ny/2-j+1)^2);
                if (dist<=Rf || dist2<=Rf)
                    base(i,j)=F;
                elseif (dist>Rf & dist<=Ri || dist2>Rf & dist2<=Ri)
                    base(i,j)=I;
                end
```

```
            end
        end

        base2=flipud(base);base12=[base;base2];
        base13=fliplr(base12);
        base14 = [base12,base13];

        Ma= base14==M;Marea=sum(Ma(:));
        Fa= base14==F;Farea=sum(Fa(:));
        Ia= base14==I;Iarea=sum(Ia(:));
        Ta=Marea+Farea+Iarea;
        fvr= Farea/Ta; ivr=Iarea/Ta;
        Rai(ii)=Ri;
        fv(ii)=fvr;
        iv(ii)=ivr;
        if (iv(ii)>=Ivr)
            break
        end
    end
    Ri=Rai(ii);
    if (ii>1)
        if ( abs(Ivr-iv(ii)) > abs(Ivr-iv(ii-1)) )
        Ri=Rai(ii-1);
        end
    end
%-------------------------------------------------------------
   % Generate current Ruc;
    base = M*ones(nx/2,ny/2); % Rectangular RUC with two of % the
opposite
        for i=1:nx/2
            for j=1:ny/2
                dist=sqrt(i^2+j^2);
                dist2=sqrt((nx/2-i+1)^2 + (ny/2-j+1)^2);
                if (dist<=Rf || dist2<=Rf)
                    base(i,j)=F;
                elseif (dist>Rf & dist<=Ri || dist2>Rf & dist2<=Ri)
                    base(i,j)=I;
                end
            end
        end
        base2=flipud(base);base12=[base;base2];
        base13=fliplr(base12);
        base14 = [base12,base13];

        Ma= base14==M;Marea=sum(Ma(:));
        Fa= base14==F;Farea=sum(Fa(:));
        Ia= base14==I;Iarea=sum(Ia(:));
        Ta=Marea+Farea+Iarea;
        fvr= Farea/Ta; mvr=Marea/Ta; ivr=Iarea/Ta;

%-------------------------------------------------------------
else
```

```
  % PMC. No interface.
    base = M*ones(nx/2,ny/2); % Rectangular RUC with two of the
opposite
        for i=1:nx/2
            for j=1:ny/2
                dist=sqrt(i^2+j^2);
                dist2=sqrt((nx/2-i+1)^2 + (ny/2-j+1)^2);
                if (dist<=Rf || dist2<=Rf)
                    base(i,j)=F;
                end
            end
        end
    base2=flipud(base);base12=[base;base2];
    base13=fliplr(base12);
    base14 = [base12,base13];
        Ma= base14==M;Marea=sum(Ma(:));
        Fa= base14==F;Farea=sum(Fa(:));
        Ta=Marea+Farea;
        fvr= Farea/Ta; mvr=Marea/Ta;
end
        RUC = base14;
```

Randomly Packed Circular Fibers RUC Generation

This function generates random architectures for RUCs with a specified number of fibers in a square packing. The subcells are of uniform size with an aspect ratio 1. The program ensures periodicity by mirroring those fibers which intersect the boundaries. The program accepts nominal inputs for fiber and interface radii, and the overall RUC size in number of subcells in width and height directions.

```
function [FVR,IVR,MVR,RUC,BigRUC,BigL,BigH,FibCenters,Ta] =...
  new_CMC_Ruc2 ( Radf,Radi, Nfibers,nrep,dpad,MaxTries,fvr,ClCntrl,ClExpo)
% Program to generate random placement of circular fiber RUC
% in a nx x ny Region...

M=2;F=1;I=3;U=5;
e=0;
[BaseRUC]=BaseCmcRuc3(Radf,Radi,e);
[Dia,~]=size(BaseRUC);
Rad=Dia/2;
FA = length ( BaseRUC(BaseRUC==F));
IA = length ( BaseRUC(BaseRUC==I));

% Find Big RUC size
Side= round( sqrt( FA*Nfibers/fvr));
MA = Side^2-Nfibers*(FA+IA);
FVR = FA*Nfibers/Side/Side;
IVR = IA*Nfibers/Side/Side;
MVR = MA/Side/Side;
nx=Side;
ny=Side;
%--------------------------------------------------------------------------
% dpad=0; % Fibers can touch
% dpad = 1, minimum of one subcell between the fibers.

    IndForI=find(BaseRUC==F | BaseRUC==I);

% Mark all the nodes that will be unavailable after placing a fiber
    ElimRUC=BaseCmcRuc3(2*Radf,2*(Radi+dpad),0.1);
    ElimRUC(ElimRUC==F | ElimRUC==I)=U;
    IndForU = find(ElimRUC==U);
%--------------------------------------------------------------------------
% All locations in ElimRUC are unavailable once a BaseRUC is placed in
% place at any location. The positions of the subcells change depending on
% the fiber center.
%--------------------------------------------------------------------------

[NsizeE,~]=size(ElimRUC);
Maxgen=MaxTries;
RUC=M*ones(nx,ny);
RUCU=RUC;
% Initialize variables

GenV=zeros(1,Maxgen);
NfibV=zeros(1,Maxgen);
Nf =zeros(Nfibers,2);  % Fiber Centers

nfib = 0;
% Define the domain for random center
```

```
xmin = 1; xmax = nx ;
ymin = 1; ymax = ny ;
%-------------------------------------------------------------------------
% Make a list of all eligible points in the domain.
NsizeB=(xmax-xmin+1);
EligRefL =cell(NsizeB^2,1);
EligIndx = (1:NsizeB^2)';
X=xmin:xmax;
Y=ymin:ymax;
l=0;

for j = 1:NsizeB
    for i=1:NsizeB
        l=l+1;
        EligRefL {l}= [X(i),Y(j), RUCU(i,j)];
    end
end
EligibleL=EligRefL;
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
Igen=1;
while ( Igen <= MaxTries && nfib < Nfibers)  % First While Loop checking
Number of fibers placed
        % Generate random center;
        GenV(Igen)=Igen;
        % generate uniform random number
        % pick a random location;
        if (ClCntrl==0)
                Rloc = randi(length(EligIndx),1);
                LocR = EligibleL{Rloc};
        elseif (nfib==0)
                Rloc = randi(length(EligIndx),1);
                LocR = EligibleL{Rloc};
        else
%       find a location based on probability of proximity to a fiber
        [LocR] = CloseLocation(EligibleL,EligIndx,Nf(1:nfib,:),ClExpo);
        end

        Cx= LocR(1,1);Cy=LocR(1,2);
        FibLoc = [Cx , Cy ];

        Nf(nfib+1,:) = FibLoc(1,:); % Store Fiber Centers
        % show square fiber for now
        px=FibLoc(1,1)-Rad;py=FibLoc(1,2)-Rad;
        Irow = px:px+Dia-1;
        Jcol = py:py+Dia-1;

        % Check if TempRUC dimension will go outside of box EJP
        [Irow,Jcol]=CheckBDcross(ny,nx,Irow,Jcol);
        tmp3 = RUC(Irow,Jcol);
        tmp3(IndForI)=BaseRUC(IndForI);
        RUC(Irow,Jcol)=tmp3;
%         figure(1), plotRUC(RUC )
        % Add code for Unavailable locations here
        pxU=FibLoc(1,1)-NsizeE/2;pyU=FibLoc(1,2)-NsizeE/2;
```

```
         IrowU = pxU:pxU+NsizeE-1;
         JcolU = pyU:pyU+NsizeE-1;
        % Check if ElimRUC dimension will go outside of box
        [IrowU,JcolU]=CheckBDcross(ny,nx,IrowU,JcolU);
         ind =  find(RUCU(IrowU,JcolU)==U);
         tmp = RUCU(IrowU,JcolU);
         tmp2 = ElimRUC;
         tmp2(ind) = tmp(ind);
         RUCU(IrowU,JcolU) = tmp2;

        % Increment the trial counter
        nfib=nfib+1;
        NfibV(Igen)=nfib;

        % Update Eligible Region by removing the fiber that is already
        % placed with in the domain
        [EligIndx,EligibleL] = PurgeAndUpdate(IrowU,NsizeE,JcolU,...
            NsizeB,EligIndx,EligRefL,ElimRUC);
        TF=isempty(EligIndx);

        if (TF && nfib < Nfibers)
            % Start the counter again
            Igen = Igen+1;
            nfib=0;
            EligIndx = (1:NsizeB^2)';
            EligibleL=EligRefL;
            RUC=M*ones(nx,ny);
            RUCU=RUC;
            Nf = zeros(Nfibers,2);  % Fiber Centers
        end
 end

% ------------------------------------------------------------------------

% Check Whether max generations reached...

if (nfib < Nfibers && Igen>MaxTries)
    Msg={'Max.Gens Reached','Failed to Place all Fibers',...
        'Decrease FVR','Or increase Radf and try again'};
    warndlg(Msg,'Failure','modal');
end
% ------------------------------------------------------------------------

Generation=GenV(1:Igen)';
N_of_fibers=NfibV(1:Igen)';
Ta= table(Generation,N_of_fibers);
Lbase=nx;Hbase=ny;
[BigRUC,BigL,BigH] = RepeatRUC (Lbase,Hbase,RUC,nrep);
FibCenters = [ Nf(:,1)-1 Nf(:,2)];



function[EligIndx,EligibleL] = PurgeAndUpdate(Irow,NsizeS,Jcol,...
                               NsizeB,EligIndx,EligRef,ElimRUC)
% Update Eligible Region by removing the locations occupied by the fibers
% that are already placed with in the domain
```

```
l=0;
for j = 1:NsizeS
    for i= 1:NsizeS
        if(ElimRUC(i,j)==5)
            l=l+1;
            SmallL(l)=Irow(i) + (Jcol(j)-1)*NsizeB ;
        end
    end
end
             %  Purging operation
             EligIndx=setdiff(EligIndx,SmallL);
             EligibleL=EligRef(EligIndx);

function[base]=BaseCmcRuc3(Radf,Radi,e)
% Program that generates a Square packing CMC Ruc with interface.
% Fiber radius and Interface thickness are user definable.
% Developed by Pappu L.N. Murthy, Date June, 30, 2015;
%
M=2;F=1;I=3;
nx=round(Radf); ny=round(Radi);
if(ny>nx)
    nx=ny;
end
base = M*ones(nx ,ny );            % Square RUC
% define center of RUC
 cx=nx;cy=0;
[X,Y]= meshgrid((1:nx)-0.5 , (1:ny)-0.5  ) ;
Dist = sqrt( (cx-X).^2 + (cy-Y).^2 )-e;
base(Dist<=Radf)=F;
 base(Dist>Radf  &  Dist<=Radi) =I;
base= [flipud([ base,fliplr(base)]);[ base,fliplr(base)]];

function[Irow,Jcol]=CheckBDcross(ny,nx,Irow,Jcol
% Check if TempRUC dimension will go outside of box EJP
% This function checks each fiber block to see if it crosses the boundary
% and if it crosses the boundary it tries to place the remaining fiber on
% the opposite side to preserve periodicity of the RUC

            for i=1:length(Irow)
                if Irow(i)>ny
                    Irow(i)=Irow(i)-ny;
                elseif Irow(i)<1
                    Irow(i)=Irow(i)+ny;
                end
            end
            for j=1:length(Jcol)
                if Jcol(j)>nx
                    Jcol(j)=Jcol(j)-nx;
                elseif Jcol(j)<1
                    Jcol(j)=Jcol(j)+nx;
                end
            end
```

Randomly Packed Square Fibers RUC Generation

This function generates a square packed disordered fibers in an RUC where the circular fibers are approximated as squares and the subcell sizes are nonuniform. The fiber size and interface thickness can be randomly distributed based on user parameters CovF, and CovI which stand for the coefficient of variation in fiber diameter and interface thickness respectively. The main advantage of such an RUC is that, for a much smaller size one can study the architectural effects using the higher order version of MAG/GMC, namely HFGMC and satisfying the fiber and interface volume requirements exactly. In case of uniform square subcells, it is not possible to satisfy exactly the fvr and ivr requirements in addition to having to deal with a very large RUC size incurring high computational costs.

```
function [Ruc,L,H,success,FVR,IVR,MVR] = ...
RectFiberPlacementWithMirroring(Nfibers,fvr,ivr,Eclr,CovF,CovI)

% Rectangular variable size fiber placement
% developed by Pappu L. N. Murthy
% Date May 16, 2016
format long;
Nsize= Nfibers*4+1;
% Variable Size Square fibers
BetaPar1 = 1; BetaPar2=1; % for Beta distribution use parameters % >1;
% for uniform distribution use parameters = 1;
CovF=.4;CovI=.4; % percentage variation in fiber diameter and
% interface thickness use a value between 0 and 1;
mind=1-CovF;maxd=1+CovF;
mini=1-CovI;maxi=1+CovI;
Int=maxd-mind;      % Max difference between diameters.
df = mind+betarnd(BetaPar1,BetaPar2,Nfibers,1)*Int;
FA = sum (  df.^2 );
dRucW  = sqrt(FA/fvr);
dRucH  = dRucW;
A  = dRucW*dRucH;
% Uniform thickness interface
IA =A*ivr;
y = @(xx)( sum((2*xx+df).^2 -df.^2) - IA);
iT = fzero(y,[0,10]); % average thickness of interface based on   %
number of
% fibers, fiber volume ratio and
% interface volume ratio
% ------------------------------------------------------------
% Variable interface thickness
% Create for (Nfibers-1) fibers random interface thicknesses.then
% calculate the remaining interface thickness based on the
% interface volume
% ratio requirement to be met
miniT=mini*iT;maxiT=maxi*iT;
flag = 0;
while (flag==0)
    iTr = miniT+betarnd(BetaPar1,BetaPar2,Nfibers-1,1)*(maxiT-miniT);
    tempA = (df(1:Nfibers-1) + 2*iTr(1:Nfibers-1) ).^2 ...
        - df(1:Nfibers-1).^2;
    LastIntA= ( IA - sum(tempA));
```

```
    if(LastIntA >0)
        y=@(xx)(2*xx+df(Nfibers)) ^2 -df(Nfibers) ^2 - LastIntA;
        flag = 1;
    else
        flag = 0;
    end
end
iTr(Nfibers) =  fzero(y,[0,10]);
idF=df+2*iTr;
% ---------------------------------------------------------
% Check
Eps=Eclr*mean(df);
% Define RUC boundary
RucX=[0,dRucW,dRucW,0,0];
RucY=[0,0,dRucH,dRucH,0];
xL = [0,dRucW]; yL= [0,dRucH];
% ---------------------------------------------------------
% define upper and lower bounds for the fiber to be placed       %
depending on
% its own dimensions:
px=0;py=0;
lb=[xL(1), yL(1)]; ub = [xL(2), yL(2)];
% ---------------------------------------------------------
%---------------------------------------------------------
% Generate Randomly placed fibers in RUC.
Ngen=10;  % Number of generations
success=0; % Flag that checks whether we are successful in
% placing fibers.
Ntry=1000; % Number of tries for each generation
tI=iTr;
for igen=1:Ngen
    Ctmp=zeros(Nfibers,2);Rtmp=zeros(Nfibers,4);
    itry=0;
    nfib=0;
    for i = 1:Nfibers
        if (itry >= Ntry)
            break
        end
        if (i == 1)
            nfib=0;
            %      Generate Fiber centers
            Ctmp(i,1)= random('unif',lb(1),ub(1));
            Ctmp(i,2)= random('unif',lb(2),ub(2));
            % Construct rectangle
            Rtmp(i,:)=[Ctmp(i,1)-idF(i)/2,Ctmp(i,2)-
idF(i)/2,idF(i),idF(i)];
            itry=itry+1;
            nfib=nfib+1;
            % Check whether the fiber cuts any boundaries
            iT=(Rtmp(i,3)-df(i))/2;
            [CB,tB,RtmpB] = BdCheckTest(Ctmp,i,lb,ub,Rtmp(i,:)
,dRucW,dRucH,iT);
```

```
            % Show the position of fibers
            Ctot=[Ctmp(1:i,:);CB ];
            iTtot=[tI(1:i,1);tB];
            Rtot= [Rtmp(1:i,:);RtmpB];
        else
            Ctmp(i,1)= random('unif',lb(1),ub(1));
            Ctmp(i,2)= random('unif',lb(2),ub(2));
            % Construct rectangle
            Rtmp(i,:)=[Ctmp(i,1)-idF(i)/2,Ctmp(i,2)-
idF(i)/2,idF(i),idF(i)];
            itry=itry+1;
            % Check whether the fiber cuts any boundaries
            iT=(Rtmp(i,3)-df(i))/2;
            [CB,tB,RtmpB] = BdCheckTest(Ctmp,i,lb,ub,Rtmp(i,:)
,dRucW,dRucH,iT);

         % first mix regular and boundary fibers into one group
            Ctemp= [Ctmp(1:i,:);Ctot(i:end,:);CB ];
            iTtemp=[tI(1:i,1);iTtot(i:end);tB];
            Rtemp= [Rtmp(1:i,:);Rtot(i:end,:);RtmpB];
            % Check for feasibility;
            [flag] = IntersectFibers(Rtemp,Eps);

            while(flag==1 && itry < Ntry)
                itry=itry+1;
                Ctmp(i,1)= random('unif',lb(1),ub(1));
                Ctmp(i,2)= random('unif',lb(2),ub(2));
                Rtmp(i,:)=[Ctmp(i,1)-idF(i)/2,Ctmp(i,2)-
idF(i)/2,idF(i),idF(i)];
                % Check whether the fiber cuts any boundaries
                iT=(Rtmp(i,3)-df(i))/2;
                [CB,tB,RtmpB] = BdCheckTest(Ctmp,i,lb,ub,Rtmp(i,:)
,dRucW,dRucH,iT);
           % first mix regular and boundary fibers into one group
                Ctemp= [Ctmp(1:i,:);Ctot(i:end,:);CB ];
                iTtemp=[tI(1:i,1);iTtot(i:end);tB];
                Rtemp= [Rtmp(1:i,:);Rtot(i:end,:);RtmpB];
                % Check for feasibility;
                [flag] = IntersectFibers(Rtemp,Eps);
            end
            if (itry <= Ntry)
                nfib=nfib+1;
                Ctot=Ctemp;
                iTtot=iTtemp;
                Rtot=Rtemp;
            end
        end
    end
    if (nfib<Nfibers)
        % not successful
        % clear all variables and start over again
        clear nfib
```

```
    else
        % Determine whether the operation has been successful
        %-------------------------------------------------------
        % Define Grid for RUC
        [lx,ly]=Grid( iTtot,Rtot);
        % Ruc creation
        Lxsort=sort(lx);  Hysort=sort(ly);
        Lxuni=uniquetol(Lxsort);Hyuni=uniquetol(Hysort);
        Lxuni  = Lxuni( Lxuni >= 0 & Lxuni <= dRucW);
        Hyuni  = Hyuni( Hyuni >= 0 & Hyuni <= dRucH);
        Lxl=length(Lxuni ); Hyl=length(Hyuni );
        %-------------------------------------------------------
        if (Lxl+1==Nsize && Hyl+1==Nsize)
            success=1;
            break
        else
            clear nfib
        end
    end
end

if (success)
    %-------------------------------------------------------
    % Define Grid for RUC
    [lx,ly]=Grid( iTtot,Rtot);
    %-------------------------------------------------------
    % Ruc creation
    Lxsort=sort(lx); Lxsort=round(Lxsort,10);
    Hysort=sort(ly); Hysort=round(Hysort,10);
    Lxuni=unique(Lxsort);
    Hyuni=unique(Hysort);
    Lxuni  = Lxuni( Lxuni >= 0 & Lxuni <= dRucW);
    Hyuni  = Hyuni( Hyuni >= 0 & Hyuni <= dRucH);
    Lxl=length(Lxuni );
    Hyl=length(Hyuni );
    L=zeros(1,Lxl+1);
    H=zeros(1,Hyl+1);
    RUC=zeros(Lxl+1,Hyl+1,2);
    for ii=1:Lxl
        if (ii ==1)
            L(ii)= Lxuni(ii);
            H(ii)= Hyuni(ii);
        else
            L(ii)=Lxuni(ii)-Lxuni(ii-1);
            H(ii)=Hyuni(ii)-Hyuni(ii-1);
        end
    end
    L(ii+1)=dRucW-Lxuni(ii);
    H(ii+1)=dRucW-Hyuni(ii);
    [xv,yv,~,~,axv,ayv] = FiberInterfaceRegions4(Rtot,iTtot );
    for i = 1:length(H)
        if (i == 1)
```

```
            istart = H(i)/2;
        else
            istart = istart+H(i-1)/2+H(i)/2;
        end
        for j=1:length(L)
            if (j == 1)
                jstart = L(j)/2;
            else
                jstart =jstart+L(j-1)/2+L(j)/2;
            end
            RUC(i,j,:)= [jstart,istart];

        end
    end
    % Assign 1 for Fiber, 2 for matrix, 3 for interface.
    Ruc=zeros(Lxl+1,Hyl+1);
    [Nftot,~] = size(Rtot);
    for i=1:Hyl+1
        for j=1:Lxl+1
            [Ruc(i,j)] = CheckConstituent (RUC(i,j,1),RUC(i,j,2),...
                xv,yv,axv,ayv,Nftot);
            A(i,j) = L(j)*H(i);
        end
    end

    % ---------------------------------------------------------
    % Compute FVR and IVR
    RucI = Ruc==3; RucF=Ruc==1;RucM=Ruc==2;
    RucIA = RucI.*A;RucIF = RucF.*A;
    RucIM = RucM.*A;
    IVR =  sum(RucIA(:))/dRucW^2;
    FVR =  sum(RucIF(:))/dRucW^2;
    MVR =  sum(RucIM(:))/dRucW^2;
% -------------------------------------------------------------
%     figure
%     showSqRuc(Ruc,px,py,H,L)
else
    msg={'Operation Unsuccessful after 10 generations',
        'and 10000 tries. Repeat the process'};
    msgbox(msg,'No SUCCESS')
end

function [flag] = IntersectFibers(R,Eps)
% Program that checks to see if any of the fibers are
% intersecting each
% other
R1(:,1:2)=R(:,1:2)-Eps;
R1(:,3:4)=R(:,3:4)+2*Eps;
A=R1 ;B=R1 ;
Ar = rectint(A,B);
Ad=diag(Ar);
area=Ar-diag(Ad);
```

```
if (any(area(:)) )
    flag=1;
else
    flag=0;
end

function[lx,ly]=Grid( iTx,R)
% The subcell sizes H and L are computed and stored in
% lx and ly. These are later used to define H and L

[Rsortx,Ix]=sortrows(R);
[Rsorty,Iy]=sortrows(R,2);
[Nfibers,~]=size(R);
% draw grid
l=0;
for i =1:Nfibers
    iT=iTx( Ix(i));
    l=l+1;
    lx(l)= Rsortx(i,1);
    l=l+1;
    lx(l)=Rsortx(i,1)+iT ;
    l=l+1;
    lx(l)=Rsortx(i,1)+Rsortx(i,3)-iT ;
    l=l+1;
    lx(l)=Rsortx(i,1)+Rsortx(i,3);
end

l=0;
for i =1:Nfibers
    iT=iTx( Iy(i));
    l=l+1;
    ly(l)= Rsorty(i,2);
    l=l+1;
    ly(l)=Rsorty(i,2)+iT ;
    l=l+1;
    ly(l)=Rsorty(i,2)+Rsorty(i,4)-iT ;
    l=l+1;
    ly(l)=Rsorty(i,2)+Rsorty(i,4);
end

function [CB,tB,RtmpB ] = BdCheckTest(C,i,lb,ub,Rtmp, W,H,tI)
% Now we have to check to see if any fiber goes outside the
% boundary.
% Bdry 1: Center(,2) < lb(,2); Center(,2)>lb(,2)
iB=0;
df=Rtmp(1,3);
dly = C(i,2)-lb(2); duy = C(i,2)-ub(2);
dlx = C(i,1)-lb(1); dux = C(i,1)-ub(1);
BdCross=[dly,duy,dlx,dux];
BdCrossY=sum ( abs(BdCross)-df/2<=0.);

switch BdCrossY
```

```
case 0 % Fiber does not cross any boundary

    CB=[];tB=[];RtmpB=[];

case 1 %  Fiber crosses sides...

    if ( abs(dly) <= df/2)
        C11(1,1)=0;C11(1,2)=H;
        CB(iB+1,:)=C(i,:)+C11(1,:);

    elseif ( abs(duy) <= df/2)
        C11(1,1)=0;C11(1,2)=-H;
        CB(iB+1,:)=C(i,:)+C11(1,:);

    elseif (abs(dlx) <= df/2)
        C11(1,1)=W;C11(1,2)=0;
        CB(iB+1,:)=C(i,:)+C11(1,:);

    elseif (abs(dux) <= df/2)
        C11(1,1)=-W;C11(1,2)=0;
        CB(iB+1,:)=C(i,:)+C11(1,:);

    end
    tB(iB+1,:)=tI;
    RtmpB(iB+1,:)=[CB(iB+1,1)-Rtmp(3)/2,...
        CB(iB+1,2)-Rtmp(3)/2,Rtmp(3),Rtmp(3)];

case 2 % FIber is crossing both borders... Ie corner;

    % Corner 1;
    if ( abs(dly)<=df/2 && abs(dlx) <=df/2)
        C11(2,1)=W;C11(2,2)=0;CB(iB+1,:)=C(i,:)+C11(2,:);
        C11(3,1)=W;C11(3,2)=H;CB(iB+2,:)=C(i,:)+C11(3,:);
        C11(4,1)=0;C11(4,2)=H;CB(iB+3,:)=C(i,:)+C11(4,:);
    elseif(abs(dly)<=df/2 && abs(dux) <=df/2)
        C11(1,1)=-W;C11(2,2)=0;CB(iB+1,:)=C(i,:)+C11(1,:);
        C11(3,1)=0; C11(3,2)=H;CB(iB+2,:)=C(i,:)+C11(3,:);
        C11(4,1)=-W; C11(4,2)=H;CB(iB+3,:)=C(i,:)+C11(4,:);
    elseif(abs(duy)<=df/2 && abs(dux) <=df/2)
        C11(1,1)=-W;C11(1,2)=-H;CB(iB+1,:)=C(i,:)+C11(1,:);
        C11(2,1)=0;C11(2,2)=-H;CB(iB+2,:)=C(i,:)+C11(2,:);
        C11(4,1)=-H;C11(4,2)=0;CB(iB+3,:)=C(i,:)+C11(4,:);
    elseif(abs(duy)<=df/2 && abs(dlx) <=df/2)
        C11(1,1)=0;C11(1,2)=-H;CB(iB+1,:)=C(i,:)+C11(1,:);
        C11(2,1)=W;C11(2,2)=-H;CB(iB+2,:)=C(i,:)+C11(2,:);
        C11(3,1)=W;C11(3,2)=0;CB(iB+3,:)=C(i,:)+C11(3,:);
    end
    tB(iB+1,:)=tI;
    RtmpB(iB+1,:)=[CB(iB+1,1)-Rtmp(3)/2,...
        CB(iB+1,2)-Rtmp(3)/2,Rtmp(3),Rtmp(3)];
    tB(iB+2,:)=tI;
    RtmpB(iB+2,:)=[CB(iB+2,1)-Rtmp(3)/2,...
```

```
            CB(iB+2,2)-Rtmp(3)/2,Rtmp(3),Rtmp(3)];
        tB(iB+3,:)=tI;
        RtmpB(iB+3,:)=[CB(iB+3,1)-Rtmp(3)/2,...
            CB(iB+3,2)-Rtmp(3)/2,Rtmp(3),Rtmp(3)];
end

function[id] = CheckConstituent (xq,yq,xv,yv,axv,ayv,Nfibers)

% Program that determines consituents in each subcell of an RUC
M=2;F=1;I=3;
% first check whether point lies in a fiber
for i =1:Nfibers
    [in] = inpolygon(xq,yq,xv(:,i),yv(:,i));
    if(in)
        break
    end
end
if (in)
    id=F;
    return
end
% check wither point lies in interface region
for i=1:Nfibers
    [in] = inpolygon(xq,yq,axv(:,i),ayv(:,i));
    if(in)
        break
    end
end
if (in)
    id=I;
    return
end
% if no success then it is Matrix
id=M;
```

Generation of a PMC RUC With Randomly Placed Fibers

This function is useful for generating RUCs with a large number of fibers where each subcell of the RUC represents a fiber. It takes fiber volume ratio, number of subcells in H and L directions as inputs and outputs RUC, the H and L vectors, and the actual fiber and matrix volume ratios realized.

```
function [RUC,H,L,fvr,mvr] = RandRuc(FVR,nH,nL)
% Program to generate Randomly placed fiber architecture RUC
% Developed by Pappu L.N. Murthy
% Multiscale and Multiphysics Modeling Branch
% Date: June 17, 2016
% format long;  clear
H=ones(1,nH); L=ones(1,nL);
R = rand(nH,nL) ;
RUC=(R>=FVR)+1;
% Compute fiber and matrix volume ratios
RucF = RUC==1;FA=sum(RucF(:));
RucM = RUC==2;MA=sum(RucM(:));
TA=nH*nL;
fvr = FA/TA;
mvr = MA/TA;
plotRUC(RUC)
```

Generation of High Density RUCs

This function enables one to reproduce RUCs of higher density where each parent subcell is subdivided into 4, 9, 16 or n^2 subcells. It takes RUC definition and density parameter (n) as input and outputs the larger repeating unit cell RUC2.

```
function[RUC2] = HDRuc(RUC,density)

% Program to generate increased size RUC given a base RUC and    %
density
% Developed by Pappu L.N. Murthy
% Multiscale and Multiphysics Modeling Branch
% Date: June 17, 2016
% format long;  clear
% Define size

[W,H]=size(RUC);
% Define Mesh density
 n=density;% level of discretization..
           %       n =1, same as the given RUC
           %       n =1, each subcell divided to 2x2=4 subcells
           %       n =3, each subcell divided to 3x3=9 subcells
           % etc etc...
for j = 1:H

    for i=1:W
        mat=RUC(j,i);
        j1=(j-1)*n;
        for jj=1:n
            j1= j1+1;
            i1=(i-1)*n;
            for ii=1:n
                i1= i1+1;
                RUC2(j1,i1)= mat;
            end
        end
    end

end
```

Post Processing Scripts

All the MATLAB postprocessing scripts for MAC/GMC analyses described under the Section 4 (4.1 through 4.4) are provided below. Typically these are the scripts that parse the MAC outputs such as the main output (*.out file) or the auxiliary output files such as xyplots, or local stress/strain response files to produce a desired output.

Extraction of RUC Stiffnesses From a MAC/GMC Output File

This function reads the main MAC output file (*.out file) and parses the composite stiffness properties and returns them in the vector A. A contains basically nine stiffness related properties as explained in the main section.

```
function[A]= ExtractRucStiffnesses(sTitle, outID)
% Program to extract stiffness information from an RUC analyses
% Developed by Pappu L.N. Murthy
% Multiscale and Multiphysics Modeling Branch
% June, 22, 2016.
N=9; % Number of RUC stiffness related properties
A=zeros(1,N);
while ~feof(outID)
    line = fgetl(outID);
    if ~isempty(strfind(line,sTitle))
        templine = fgetl(outID); %#ok empty Line
      for m = 1:N
       sTemp =(fgetl(outID));
        d = textscan(sTemp,'%s %f');
       A(m)=d{2};
      end
      break
    end
end
```

Extraction of Laminate Stiffnesses From a MAC/GMC Output File

Similar to the previous script file, here the function parses and output the four laminate level stiffness properties.

```
function[A]= ExtractLaminateStiffnesses(sTitle, outID)
% Program to extract stiffness information from a Laminate        %
analyses
% Developed by Pappu L.N. Murthy
% Multiscale and Multiphysics Modeling Branch
% June, 22, 2016.
N = 4; % Number of Laminate Stiffnesses to be extracted
A=zeros(1,N);
while ~feof(outID)
    line = fgetl(outID);
    if ~isempty(strfind(line,sTitle))
        templine = fgetl(outID); %#ok empty Line
      for m = 1:N
       sTemp =(fgetl(outID));
        d = textscan(sTemp,'%s %f');
       A(m)=d{2};
      end
      break
    end
end
```

Extraction of Number of Cycles to Failure From MAC/GMC Output

This function takes the main output file of MAC that involves fatigue analysis and outputs the number or cycles to failure and under a given fatigue loading condition. Additionally the output also contains the execution time for the analysis.

```
function[N,CPU]=Ncycle(Title,outid)
% This function Extracts from MAC output file for a fatigue analysis
run
% the number of cycles to failure and the execution time in seconds.
% Developed by Pappu L.N. Murthy, Date: July 11, 2016,
% Multiphysics and Multiscale Modeling Branch
% NASA Glenn Research Center
frewind(outid)
C1 = textscan(outid,'%s','Delimiter','\n');
C1 = C1{1};
for i=1:length(C1)
    k= regexpi(C1{i},Title);
    if( ~isempty(k))
%         N = str2double (cell2mat(regexp( C1{i},'\d+','match')));
      break;
    end
end
 if (i==length(C1))
    % No cycle information. Load Blocks Exceeded.
    N=-1;
 else
     N = str2double (cell2mat(regexp( C1{i},'\d+','match')));
 end;
if(isnan(N))
    N=1e9;
elseif(N==inf)
    N=1e9;
end

% Read CPU times
if (N==-1)
    CPU=-1;
else
    r=extract_numbers(C1{end-1});
    n=length(r);
    if (n==1)
        CPU=r(1);
    elseif (n==2)
        CPU=r(1)*60+r(2);
    elseif (n==3)
        CPU=r(1)*3600+r(2)*60+r(3);
    else
        CPU=-1;
    end
end
```

```
fclose(outid);
function r = extract_numbers(s)
%% Replaces commas for spaces as delimiters
n=strrep(s,',',' ');

%% Splits the string as a cell array
n=regexp(n,'\s+','split');

%% Determines which cells are numbers
[n,nc,ne]=cellfun(@(x) sscanf(x,'%f'),n,'uni',false);

%% Indexes the cells that are numbers
ix=cellfun(@(x,y) x==1&&isempty(y),nc,ne);

%% Places the numbers identified in the result "r"
r=n(ix);

%% Transforms the cell array into a numeric array
r = cell2mat(r);
format shortG;
```

First Matrix Cracking Strength and Ultimate Strength for a CMC Material

This script takes the xy plot information of stress versus strain to compute the firs ply matrix cracking strength and the ultimate strength of a CMC composite system.

```
function [Pls,Us]= PlsAndUs(Fname,E,offset,stp)
%-------------------------------------------------------------------
% this function calculates the first matrix cracking or proportion
limit
% strength, and the ultimate strength.
% Developed by P. L. N. Murthy
% Multiscale and Multiphysics Branch
% NASA GRC.
% Date March, 23, 2016
%-------------------------------------------------------------------
xx=load(Fname);
flag=0.;i=0;
while (flag==0)
    i=i+1;
    x(i)=offset+i*stp;
    y(i)=E*i*stp;
    yact(i)= interp1(xx(:,1),xx(:,2),x(i));
    if (y(i) > yact(i))
        flag=1;
    else
        flag=0;
    end
end
   Pls =     ( y(i)+y(i-1))/2;
   [Us]=max(xx(:,2));
```

Local Stress Response

This function parses and outputs the local stresses in the RUC which may be used for example to produce local stress contours.

```
function[sig11,sig22,sig33,sig23,sig13,sig12,sigmean,sigeff,X,Y,RUC]
=...
    ExtractStresses(fname)

% -- Set average = 1 to perform averaging of tractions across subcell
%    boundaries.  This reproduces the average sense imposition of the
%    traction continuity conditions in HFGMC.
% -- Set average = 0 for no averaging.  Use average = 0 for GMC.
average = 0;

% -- Set nint = number of integration points in HFGMC when average = 1
if average > 0
    nint = 11;
end
% -- Load x1.dat and x2.dat files and determine their sizes

x2=load ([fname,'_x2.dat']);
m = length(x2);

x3=load ([fname,'_x3.dat']);
n = length(x3);

% -- Create grid
[X,Y] = meshgrid(x3,x2);

% -- Load sig.dat file

fid1=fopen([fname,'_sig.dat'],'r');

d1=textscan(fid1,'%f %f %f %f %f %f %f %f %f','Headerlines',1);
fclose (fid1);

DBlock = m*n;
sig=zeros(DBlock,8);
for i=1:DBlock
    for j=1:8
        sig(i,j)= d1{j}(i);
    end
end


sig11 = (reshape(sig(:,1),n,m))';
sig22 = (reshape(sig(:,2),n,m))';
sig33 = (reshape(sig(:,3),n,m))';
sig23 = (reshape(sig(:,4),n,m))';
sig13 = (reshape(sig(:,5),n,m))';
sig12 = (reshape(sig(:,6),n,m))';
```

```
sigmean = (reshape(sig(:,7),n,m))';
sigeff =   (reshape(sig(:,8),n,m))';
% Extract RUC
    fid2=fopen([fname,'_epsp.dat'],'r');
    d2=textscan(fid2,'%f %f %f %f %f %f %f %f %f','Headerlines',1);
    Mnum=zeros(DBlock,1);
    for i=1:DBlock;
        Mnum(i,1)=d2{8}(i);
    end
    RUC=(reshape(Mnum,n,m))';

%-----------------------------------------------------------------------
% Section to average boundary tractions
%-----------------------------------------------------------------------

if average > 0

    Nbeta =length(x2)/nint;
    for j=1:n
        for i=1:Nbeta-1;
          sig22(nint*i,j) = 0.5*(sig22(nint*i,j) + sig22(nint*i+1,j));
          sig22(nint*i+1,j) = sig22(nint*i,j);
          sig23(nint*i,j) = 0.5*(sig23(nint*i,j) + sig23(nint*i+1,j));
          sig23(nint*i+1,j) = sig23(nint*i,j);
        end
    end

    Ngama = length(x3)/nint;
    for i=1:m
        for j=1:Ngama-1;
            sig33(i,nint*j) = 0.5*(sig33(i,nint*j) +
sig33(i,nint*j+1));
            sig33(i,nint*j+1) = sig33(i,nint*j);
            sig23(i,nint*j) = 0.5*(sig23(i,nint*j) +
sig23(i,nint*j+1));
            sig23(i,nint*j+1) = sig23(i,nint*j);
        end
    end

end
% -- End of file
```

```
function[]=Plot_Contours(Type,X,Y,Data,Titl)
switch Type
    case '2-D'
    pcolor( X, Y,Data), shading interp;
    c=colorbar;
    c.FontSize=12;
    c.FontWeight='bold';
    title(Titl,'FontSize',18);
    xlabel('\bfx_3');
    ylabel('\bfx_2','rotation',0);
    set (gca,'FontSize',16,'FontWeight','bold','LineWidth',2)
    axis image;
    axis off;
    case '3-D'
% ------------------------------------------------------------------
-----
% 3-D Contours
    surf(X,Y,Data)
    title(Titl,'FontSize',18);
    xlabel('\bfx_3');
    ylabel('\bfx_2','rotation',0);
    set (gca,'FontSize',16,'FontWeight','bold','LineWidth',2)
    box on;
end
```

Local Strain Response

This function parses and outputs the local strains in the RUC which may be used for example to produce local strain contours.

```
function[Eps11,Eps22,Eps33,Eps23,Eps13,Eps12,X,Y] =...
    ExtractStrains(fname)

% -- Load x1.dat and x2.dat files and determine their sizes

x2=load ([fname,'_x2.dat']);
m = length(x2);

x3=load ([fname,'_x3.dat']);
n = length(x3);

% -- Create grid
[X,Y] = meshgrid(x3,x2);

% -- Load eps.dat file

fid1=fopen([fname,'_eps.dat'],'r');

d1=textscan(fid1,'%f %f %f %f %f %f','Headerlines',1);
fclose (fid1);

DBlock = m*n;
Eps=zeros(DBlock,6);

for i=1:DBlock
    for j=1:6
        Eps(i,j)= d1{j}(i);
    end
end
Eps11 = (reshape(Eps(:,1),n,m))';
Eps22 = (reshape(Eps(:,2),n,m))';
Eps33 = (reshape(Eps(:,3),n,m))';
Eps23 = (reshape(Eps(:,4),n,m))';
Eps13 = (reshape(Eps(:,5),n,m))';
Eps12 = (reshape(Eps(:,6),n,m))';
end
% -- End of file
```

Miscellaneous MATLAB Recipes for MAC/GMC Runs

All the MATLAB miscellaneous scripts for MAC/GMC analyses described under the Section 5 (5.1 through 5.3) are included here.

Monte Carlo Simulations

This script prepares input files and runs them multiple times each time varying the definition of RUC randomly, in order to produce a database of composite properties for property variability study.

```
% Program for composite property variability study via Monte-Carlo
runs
format long;
clear;

Fname='RefFile-RUC.MAC' ; % Reference input file that gets
                  % modified for each Monte Carlo simulation
CL1=[18:57];% lines in Ref file to be replaced on the fly for RUC
NSimuls = 100;  % Number of Monte - Carlo Simulations
fid = fopen('DBase.txt','wt'); % Open a text file to write all
                               % the results
Fmt= [repmat('%7.2f ', 1, 6),'\n'];

for i  = 1: NSimuls
  RUCFilName=['RUC',num2str(i),'.txt']; % File containing *RUC
                                        % block details
 [A,Pls,Us]= ModifyRefFile(Fname,RUCFilName,CL1,i);
  fprintf(fid,Fmt,A, Pls,Us);
end
fclose(fid);


function [A,Pls,Us]=ModifyRefFile(Fname,RUCName,CL1,N)
% open the GMC input file and change the parameters...
%-------------------------------------------------------------------
fileID = fopen(Fname,'r');
C1 = textscan(fileID,'%s','Delimiter','\n');
C1=C1{1};
fileID2 = fopen(RUCName,'r');
C2 = textscan(fileID2,'%s','Delimiter','\n');
C2 = C2{1};
%-------------------------------------------------------------------
%  Modify Records
C1(CL1(1):CL1(end))= C2(6:end);
% C1(CL2(1):CL2(end))= C2(6:end);
% C1(CL3(1):CL3(end))= C2(6:end);
% C1(CL4(1):CL4(end))= C2(6:end);
%-------------------------------------------------------------------
fclose(fileID);
runNumber= ['MCInp',num2str(N)];
fmac=[runNumber,'.mac'];
fout= [runNumber,'.out'];
```

```
newfileID = fopen(fmac,'wt');
t0 = tic;
while (newfileID == -1) && (toc(t0) < 120)
     pause(1);
     newfileID = fopen(fmac, 'wt');
end
for j = 1:length(C1)
 fprintf(newfileID,'%s\n',C1{j});
end
fclose(newfileID); clear C1 C2;
% % Run MAC Program!
 mac =['mac203Ver5 ', runNumber,' > tmp.out'];
dos(mac);
outId = fopen(fout,'r');
sTitle='Effective Engineering Moduli';
[A]= ExtractRucStiffnesses(sTitle, outID);
 Fname='Stress-Strain_macro.data';
stp=0.005/1000;
offset=0.005/100;
[Pls,Us]= PlsAndUs(Fname,A(4),offset,stp);
fclose(outId);
delete('tmp.out', fmac, fout, 'Stress-Strain_macro.data');
```

Fatigue Response: SN Curves

This script prepares input files and runs them multiple times each time varying the fatigue loading block in order to produce a complete SN Curve for a specific RUC or a Laminate configuration.

```
format long; clear;
InF ='NewFatGMC.MAC';
S  = [70:-5:20];
Prog= 'mac4z-3_2';
[N] = SNFatigueCurves(InF,S,Prog);

% figure
hold on;
plot(N,S,'-
.or','MarkerFaceColor','w','MarkerEdgeColor','k','LineWidth',1.5)
xlabel('# of cycles');ylabel('Fatigue Load')
set(gca,'FontSize',16,'FontWeight','bold')

function [N] = SNFatigueCurves(InF,S,Prog)
% This function generates S curves from base input file given the
loads
% array S;
% Developed by Pappu L.N. Murthy
% MultiScale and MultiPhysics Modeling Branch
% NASA Glenn Research Center, Cleveland, OH
% Date: Nov, 1, 2016;

% InF is the reference input file. The load levels are changed in this
% input file and MAC is repeated run to capture the number of cycles
to
% failure under a load level specified in "slevels'
% slevels Array containing the load levels
% Ln line # where the load block is specified and needs to be changed
for
% every run
fprez ='Run';
Title ='TOTAL NUMBER OF CYCLES =';
N = zeros(length(S),1);
MetStr='MAG='; % String where load block is specified
for iii = 1:length(S)
    Rn=[fprez,num2str(iii)];
    % Load the contents of existing file into the cell array C1;
    fileID = fopen(InF,'r');
    C1 = textscan(fileID,'%s','Delimiter','\n');
    C1=C1{1};
    for i = 1:length(C1)
         k= regexpi(C1{i},MetStr);
         if( ~isempty(k))
             Ln=i;
         break;
         end
    end
```

```
        Str =textscan(C1{Ln},'%s');
        Str=Str{1};
        nL=length(Str);
        xx='';
    for i = 1: nL
        TF = strcmp(Str{i}(1:4),MetStr);
        if (TF)
           Str{i}=[MetStr,'0,',num2str(S(iii)),',',num2str(-
S(iii)),',0' ];
        end
        xx= [xx,Str{i},' '];
    end
    C1{Ln}=xx;
    fclose(fileID);
%-------------------------------------------------------------------------
-----
    fmac= [Rn,'.mac'];
    fout= [Rn,'.out'];
    newfileID = fopen(fmac,'wt');
    len = length(C1);
        for j = 1:len
        fprintf(newfileID,'%s\n',C1{j});
        end
    fclose(newfileID); clear C1;
%-------------------------------------------------------------------------
-----
    % Run MAC Program!
    mac=[Prog,' ',Rn, '> tmp.out'];
    dos(mac);
    % Open the Output Files
    outID=fopen(fout, 'r');
    [N(iii),~]=Ncycle(Title,outID);
    delete ( fmac, fout, 'tmp.out' ,[Rn,'_dam.data'])
end

%-------------------------------------------------------------------------
-----
```

Creation of Circular Randomly Distributed Fiber RUCs for Monte Carlo Simulations

This scripts generates random microstructure RUCs of a specified number of fibers, fvr, and ivr that may be utilized in Monte Carlo simulation studies for property or fatigue response.

Matlab Script file "CreateCMC_RUCs_ForMCRuns.m"

```
% Program that creates RUCs for Monte-Carlo runs
format long;
clear;
% ------------------------------------------------------------------
% Set up RUC's for 100 MC runs.
% Control Params for RUC
Radf=6;           % Fiber Radius
Radi=7.2;         % Interface Radius
nx=40;ny=40;      % Number of subdivisions
Nfibers=4;        % Number of fibers
nrep=1;           % Nuber of reps for tiling. Not used at all
                  % here.
NSimuls = 100;    % Number of MC runs
Fn='RUC';         % Generic prefix for file name
% ------------------------------------------------------------------
for i = 1: NSimuls
Fname = [Fn,num2str(i),'.txt'];
fid=fopen(Fname,'wt');
[FVR,IVR,MVR,RUC,~,~,~] = CMC_Ruc ( Radf,Radi,nx, ny, Nfibers,nrep)
WriteSmallRuc(RUC,Fname)
fclose(fid);
end
```

Creation of Randomly Distributed Square Fiber RUCs for Monte Carlo Simulation

This scripts generates random microstructure RUCs that are square packed with square fibers of a specified number of fibers, fvr, and ivr that may be utilized in Monte Carlo simulation studies for property or fatigue response.

Matlab Script File "CreateCMC_RUCs_withsqFibers_MCRuns.m"

```
% Program that creates RUCs for Monte-Carlo runs
% Created on May 17th, 2016
% Developed by Pappu L. N. Murthy
% Multiscale and Multiphysics Modeling Branch
format long;
clear;
% ------------------------------------------------------------------
% Set up RUC's for 100 MC runs.
% Control Params for RUC
fvr    = .28;    % Fiber Volume Ratio
ivr    = 0.13;   % Interface Volume Ratio
Nfibers=4;       % Number of fibers
nrep=1;          % Nuber of reps for tiling. Not used at all
                 % here.
Nsimuls = 100 ;  % Number of MC runs
Fn='RUCSq';      % Generic prefix for file name
Eclr=0.01;       % Interfiber clearance.
% ------------------------------------------------------------------
for i = 1: Nsimuls
Fname = [Fn,num2str(i),'.txt'];
fid=fopen(Fname,'wt');
% [FVR,IVR,MVR,RUC,~,~,~] = CMC_Ruc ( Radf,Radi,nx, ny, Nfibers,nrep)
[RUC,L,H,success] = ...
RectFiberPlacementWithMirroring(Nfibers,fvr,ivr,Eclr)
WriteRandSqRuc (RUC,Fname,H,L)
fclose(fid);
end
```

References

1. Aboudi, J., Arnold, S.M., and Bednarcyk, B.A. (2013) *Micromechanics of Composite Materials A Generalized Multiscale Analysis Approach*. Elsevier, New York.
2. Bednarcyk, B.A. and Arnold, S.M. (2002) "MAC/GMC 4.0 User's Manual—Keywords Manual," NASA/TM—2002-212077/Vol. 2.
3. Bednarcyk, B.A. and Arnold, S.M. (2002) "MAC/GMC 4.0 User's Manual—Example Problem Manual," NASA/TM—2002-212077/Vol. 3.
4. Pappu, L.N. Murthy and Evan J. Pineda, "Tool for Generation of MAC/GMC Representative Unit Cell for CMC/PMC Analysis," NASA/TM—2016–219127, September, 2016.
5. Arnold, S.M.; Mital, S.K.; Murthy, P.L.N.; and Bednarcyk, B.A.: "Multiscale Modeling of Random Microstructures in SiC/SiC Ceramic Matrix Composites within MAC/GMC Framework", Proc. of 31st Annual Technical Conference, American Society of Composites, Williamsburg, Virginia, Sep. 19-21, 2016.
6. Mital, S.K.; Arnold, S.M.; Murthy, P.L.N.; and Bednarcyk, B.A.:" Micromechanics-based Modeling of Random Microstructures in SiC/SiC Ceramic Matrix Composites using MAC/GMC Computer Code", 41st Annual Conference on Composites, Materials and Structures (Restricted Sessions), Cocoa Beach, Florida, Jan. 23-26, 2017.
7. Arnold, S.M.; Murthy, P.L.N.; Bednarcyk, B.A.; Pineda, E.J.; and Mital, S.K.:" Micromechanics-Based Fatigue Life Prediction of Composites", Proc. of 2017 AIAA SciTech Conference, Grapevine, Texas, Jan. 9-13, 2017.
8. Goldberg, R.K., Comiskey M.D., and Bednarcyk, B.A. (1999) "Micromechanics Analysis Code Post-Processing (MACPOST) User Guide, Version 1.0, NASA/TM—1999-209062.